

第1章　MATLAB是什么

没有MATLAB就没有乐趣。

Nachtigal, M. N., Reddy, S. C., Trefethen, L.N.(1990)。
不对称矩阵迭代有多快？
关于迭代方法的Copper Mountain会议论文集，
Copper Mountain CO, 1-5,1990年4月。

1.1　MATLAB能做什么

MATLAB是一个可视化的计算程序，被广泛地使用于从个人计算机到超级计算机范围内的各种计算机上。

MATLAB包括命令控制、可编程，有上百个预先定义好的命令和函数。这些函数能通过用户自定义函数进一步扩展。

MATLAB有许多强有力的命令。例如， MATLAB能够用一个单一的命令求解线性系统，能完成大量的高级矩阵处理。

MATLAB有强有力的二维、三维图形工具。

MATLAB能与其他程序一起使用。例如， MATLAB的图形功能，可以在一个 FORTRAN程序中完成可视化计算。

25个不同的MATLAB工具箱可应用于特殊的应用领域。

MATLAB在以下的领域里解决各种问题是一个十分有效的工具：

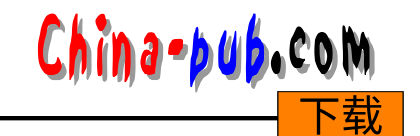
• 工业研究与开发。
• 数学教学，特别是线性代数。所有基本概念都能涉及。
• 在数值分析和科学计算方面的教学与研究。能够详细地研究和比较各种算法。
• 在诸如电子学、控制理论和物理学等工程和科学学科方面的教学与研究。
• 在诸如经济学、化学和生物学等有计算问题的所有其他领域中的教学与研究。
• 在MATLAB中创建的组是矩阵， MATLAB的名字取自矩阵实验室(MATrix LABoratory)。

1.2　MATLAB实例

本节中的实例恰当而简洁地展示了 MATLAB能做什么。在一些实例中给出了完整的MATLAB命令；而在另一些实例中，为简化仅给出部分命令。

在本书中出现的 MATLAB代码用的是一种特殊的字体以区别于书中别的文字。MATLAB的输出是斜体字，即：我们输给 MATLAB的命令是正体；MATLAB给出的输出答案是斜体。

百分符号%在MATLAB中用做注释符号，在本书中全部都是这样使用。采用的其他表示方法是：数量和预定义函数用斜体字，矩阵、向量和用户自定义函数用黑体字。矩阵用大写



字母开头命名，而向量以小写字母开头。细胞矩阵是如同矩阵或向量的概念，也采用黑体字，其结构和对象也是如此。在命令表中，用斜体字表示那些可选的函数参数。例如，command(parl, *par2*)，参数par1总是需要的，而*par2*是可选的。

■ 例1.1 二维和三维函数

MATLAB能用于计算，并以二维和三维图形显示各种函数。在 MATLAB函数中包括了所有主要的数学函数和大量的高级函数。

(a) 用简短的MATLAB命令计算并绘制在$0\le x\le 6$范围内的$\sin(2x)$、$\sin x^2$和$\sin^2 x$。

```
x=linspace(0,6);  % 创建一个向量x。
y1=sin(2*x);      % 向量y1等于x坐标上某一x的sin(2x)值。
y2=sin(x.^2);     % 向量y2等于sin(x.^2)，同上。
y3=(sin(x)).^2;   % 向量y3等于(sin(x)).^2，同上。
```

命令plot(x,y1)绘制向量**y1**，**y1**作为向量**x**的一个函数，plot命令的定义可参见第13章。由此能够很容易地在一个图上绘制$\sin(2x)$、$\sin(x^2)$和$\sin^2 x$的曲线并正确地标记它们(图1-1)。

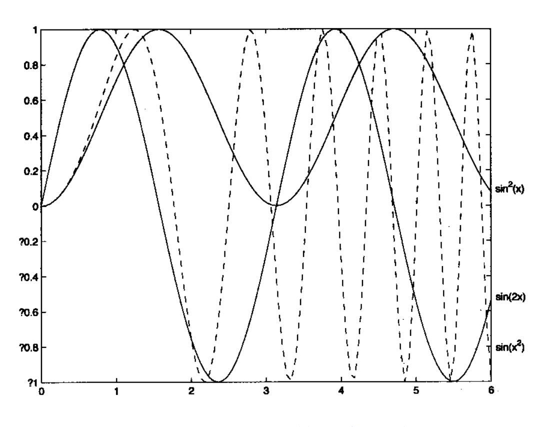


图1-1 同一图上的三条曲线

(b) 两个变量的函数需要用三维来恰当地图示，MATLAB能够给出很好的三维图。在图1-2中，用四个不同的方法展示了函数$f(x, y) = \cos(x)\cdot\sin(y)$的图形：左上图用surf命令和shading interp；右上图用mesh；左下图用waterfall以及右下图用contour。关于图像命令的详细信息可参见第13章。

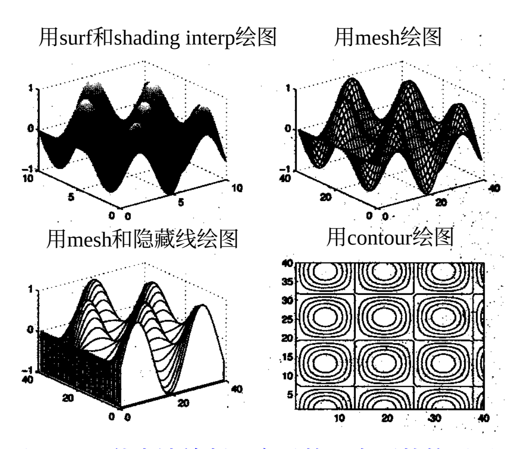
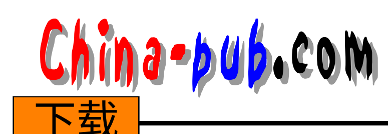


图1-2 四种方法绘制双变量的一个函数的图形



(c) MATLAB也能绘制一条参数曲线，例如：

$$\begin{cases} x = \cos t - \sin 3t \\ y = \sin t \cos t - \cos 3t \end{cases}$$

x-y平面图如图1-3所示。

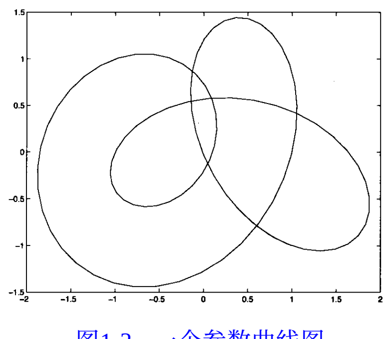


图1-3 一个参数曲线图

■

■ **例1.2 函数分析**

MATLAB命令`fzero`和`fmin`可以用于寻找一个函数的零点和最小值。

函数 $xe^{x^2} - e^{x^2} - \sin x^3$ 可以用名叫 **func**的用户自定义函数 (见2.9节)表示，并存入一个名叫**func.m**的M文件中。这个文件由下列行组成：

```
function y = func(x)
y = x.*exp(x.^2) - exp(x.^2) - sin(x.^3);
```

如果这个M文件被存放在当前的工作目录中，或在一个称为 **matlab**的子目录中，函数**func**就可以像预定义的MATLAB函数一样调用。例如，调用`xiszero=func(0)`，给出的答案是：

```
xiszero =
      -1
```

用这样定义的函数， MATLAB提供了一个命令来寻找方程 $xe^{x^2} - e^{x^2} - \sin x^3 = 0$的零点。命令`xsolv=fzero('func', 3)`给出：

```
xsolv =
     1.2194
```

在本例中，命令中的第2个自变量用的是3，是开始计算的一个初始近似值。

如果在$-1 \le x \le 1.5$区间内绘制这个函数，则正确答案如图 1-4所示。

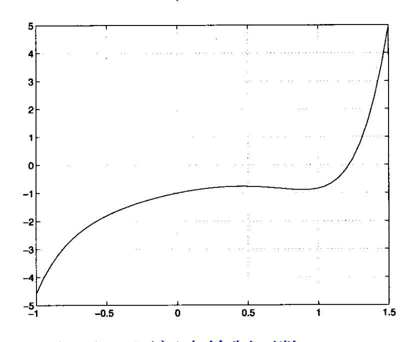


图1-4 在$-1 \le x \le 1.5$区间内绘制函数$xe^{x^2} - e^{x^2} - \sin x^3$的图形



当x在0.5和1之间时，这个函数看起来有一个极小值，为正确找出这个极小值，用命令 mpoint=fmin('func', 0.5, 1) 其结果为：

```
mpoint =
      0.8954
```

用于检查MATLAB中用户自定义函数的命令可参见第10章和第11章。

■

■ 例1.3　线性系统与特征值

(a) MATLAB可以用一个简单的命令行求解线性系统，系数矩阵 **A**和右侧**b**定义如下：

$$\mathbf{A}=\begin{pmatrix}3 & 1 & -1\\ 1 & 2 & 4\\ -1 & 4 & 5\end{pmatrix}\quad \mathbf{b}=\begin{pmatrix}3.6\\ 2.1\\ -1.4\end{pmatrix}$$

这对应于线性系统**Ax=b**，如下所示：

$$\begin{cases}3x_1+x_2-x_3=3.6\\ x_1+2x_2+4x_3=2.1\\ -x_1+4x_2+5x_3=-1.4\end{cases}$$

这可由如下命令求解：

```
x=A\b
```

其结果为：

```
x =
    1.4818
   -0.4606
    0.3848
```

(b) 还有许多矩阵控制命令。例如，例(a)中矩阵A的特征值很容易地可以由下列命令得到：

```
[EigenVectors,EigenValues] = eig(A)
```

其给出：

```
EigenVectors =
   -0.9482   -0.3129   -0.0553
   -0.2887    0.7756    0.5613
    0.1328   -0.5482    0.8258

EigenValues =
    3.4445         0         0
         0   -1.2305         0
         0         0    7.7860
```

矩阵**EigenVectors**的列是**A**的特征向量，**EigenValues**中对角线元素是特征值。由于矩阵 **A** 是对称的，因此，所有的特征值都是实数，三个特征向量是相互正交的。

■

MATLAB中的基本概念是矩阵。基本的矩阵命令在第3章描述，更多的命令将在第4、7、8、9章中描述。

■ 例1.4　曲线拟合与插值

(a) 如果有两个向量**x**和**y**表示的x-y平面上的一组点，那么，可以对它们进行插值点或者拟合一条曲线。令

$$\mathbf{x}=(1\quad 1.5\quad 3\quad 4\quad 5\quad 6\quad 6.5\quad 7\quad 8)$$



y =(1.2 1 1.7 2.5 2 2.3 2.5 3 3.1)

对应*x*-*y*平面上的9个点。首先，展示以最小二乘法拟合数据的线性函数，这个可以通过MATLAB中的三个简单的命令来实现：

```
p1=polyfit(x, y, 1)        % p1=A向量等于一次多项式的系数。
linc=polyval(p1, x)        % linc=A向量等于x点上多项式p1的值。
plot(x, linc,x, y,'x')     % 绘制多项式和由'x'标记的数据。
```

结果见图1-5(左图)。

能以最小二乘法对一组点拟合高次多项式。对上面的命令行进行一点小改动就可以得到 7次多项式：

```
p7=polyfit(x, y, 7);        % p7=A向量等于7次多项式的系数。
xx=1:0.25:8;                % xx=所有想要进行多项式计算的点。
polc=polyval(p7, xx);       % polc=A向量等于点xx上多项式p7的值。
plot=(xx,polc, x, y,'x')    % 绘制多项式和由'x'标记的数据。
```

其结果如图1-5(右图)所示。

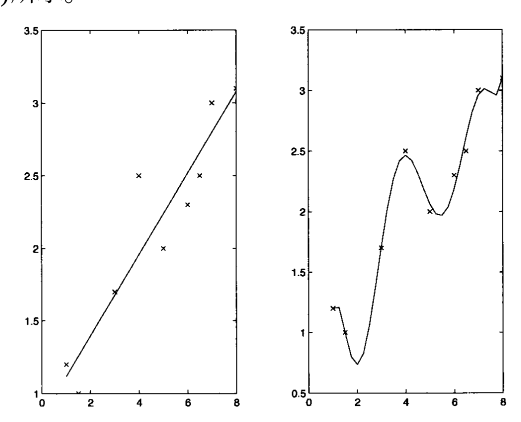


图1-5 *x*-*y*平面上对一组具有9个点的数据拟合的1次和7次多项式

(b) MATLAB提供了二维和三维的内插函数。给定一组点 (x_i, y_i)和一些内插点$\tilde{x}_i$,MATLAB能返回通过对这些数据内插的插入点的值，这可以有不同的方法实现。作为一个例子，将使用(a)中的一组点来给出在下列点中插入的值：

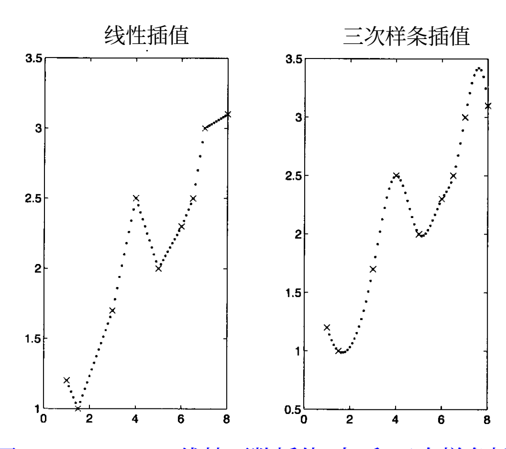


图1-6 piecewise 线性函数插值(左)和三次样条插值(右)

$\tilde{x}_i$=(1 1.1 1.2 1.3 ··· 7.9 8)

在图1-6中，分别展示了分段线性插值和三次样条插值。‘ x ’标记表示原始数据，点线是中间点的被插函数。

有关插值和曲线拟合的详细信息可参见第 10章。

■

■ 例1.5 统计

MATLAB包含了统计命令。例如，很容易地求得实验数据的平均值和中位值以及绘制统计频数直方图或直方图。

图1-7显示了某小镇上每个人的年龄。上图是统计频数直方图，显示了每个年龄的人数。例如，看到两个年龄最大的人是 92岁。统计频数矩形图也表示了小镇上没有人是 11岁或12岁，7岁的儿童7人。

也能看出32岁以下的人与32岁以上的人一样多，因为这是一个中间年龄。此外，平均年龄是35岁。这些也都在图中标出。

另外，下图展出了小镇上居民的年龄。例如，如果知道要列出的是第 11人，那么也就知道这个人是一个小孩。因为第 11个直方块接触*x*轴。这种绘图称为阶梯图，即没有内部线的直方图。统计命令在第6章中描述。

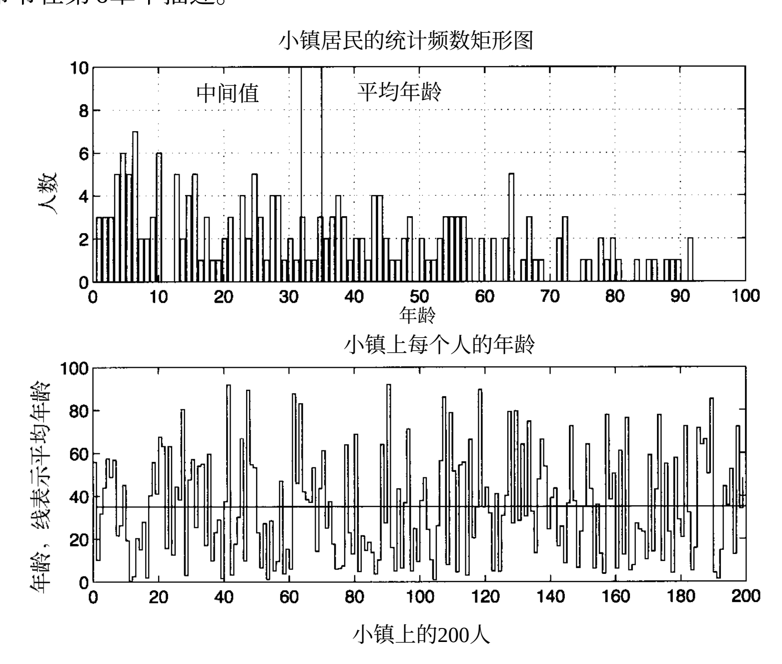


图1-7 来自小镇数据的一个统计频数矩形图

■

■ 例1.6 傅立叶变换与信号分析

MATLAB能利用快速傅立叶变换**FFT**计算离散傅立叶变换，这用于信号分析和解微分方程。为了证明MATLAB的傅立叶变换，用0和1之间的随机数干扰5sin(x)+2sin(5x) 函数:

```
x=linspace(0,*2pi, 64);
signal=5*sin(x)+2*sin(5*x)+randn(x);
```

干扰信号和原始信号如图1-8(上)所示。

然后，变换这个信号，并且删除变换后信号中的所有高频，即系数向量中心部分设为零。

```
transf=fft(signal);
filttransf(1:9)=transf(1:9);
filttransf(56:64)=transf(56:64);
```

傅立叶变换的实际部分如图1-8(下)所示。图1-9(上)中删除了高频傅立叶变换，仅低频向量通过逆傅立叶变换而变换：

```
filtsig=ifft(filttransf);
```

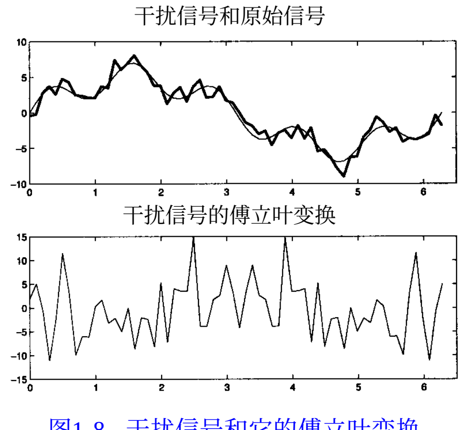


图1-8 干扰信号和它的傅立叶变换

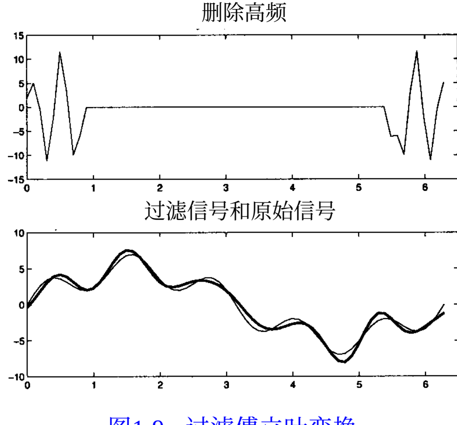


图1-9 过滤傅立叶变换

这个过滤的信号与原始信号一起在图1-9(下)中展示。这个过滤的信号如期望一样是光滑的。因为这种干扰也影响部分低频信号，所以它与原始信号不等。

在10.5节中，作为MATLAB的一部分描述了傅立叶变换，更详细的有关信号处理工具箱的信息参见附录C。

■

■ 例1.7 常微分方程

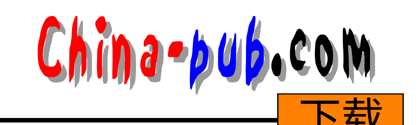
MATLAB可以用数字求解常微分方程。作为一个实例，展示在一个轨道上一只猫追赶一只机器玩具老鼠。猫和老鼠的位置分别用 $(x(t), y(t))$和$(X(t), Y(t))$表示。猫的速度向量是和猫与老鼠之间的差向量对应的。猫从(50，40)开始，这就给出了如下的微分方程系统：

$$\begin{cases} \dfrac{dx}{dt} = \dfrac{\omega}{\alpha}(X(t) - x(t)) \\ \dfrac{dy}{dt} = \dfrac{\omega}{\alpha}(Y(t) + y(t)) \\ x(0) = 40 \\ y(0) = 50 \end{cases}$$

式中w是老鼠的速度，a是猫和老鼠在t时刻相距的距离，由下式给出：

$$\alpha(t) = \sqrt{(X(t) - x(t))^2 + (Y(t) - y(t))^2}$$

这个系统的解如图1-10所示，从图中可以看到由于老鼠跑得太快，猫并没有成功地抓到老鼠。



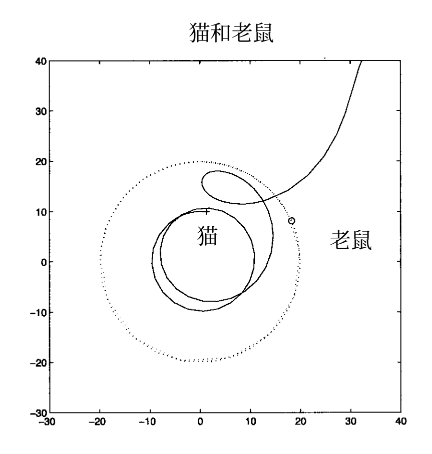


图1-10　猫捉机器玩具老鼠。小圆是老鼠的终点，十字是猫的终点

在11.2节中描述了如何使用MATLAB求解一般的常微分方程问题。

■

■ 例1.8　偏微分方程

在MATLAB中有许多用于不同应用领域的工具箱。这里，作为一个例子给出 PDE工具箱，这个工具箱能通过使用有限元方法 (the Finite Element Method)解椭圆方程、抛物线方程和双曲线方程。区域要分成大量的三角形子区域，对每个三角形，其解由一个简单函数估计。所用的三角形越多，其偏差就越小。

求解这个区域里的椭圆问题－$\Delta u=\sin(2\pi y+\pi/2)\cos(2\pi x+\pi/2)$，如图1-11所示。

拉普拉斯算子Δ是$\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}$，边界条件$u$=0，也可由PDE工具箱所做的三角形子区域给出。

这个问题的解可以用可视化等高线以二维绘制，如图 1-12所示。

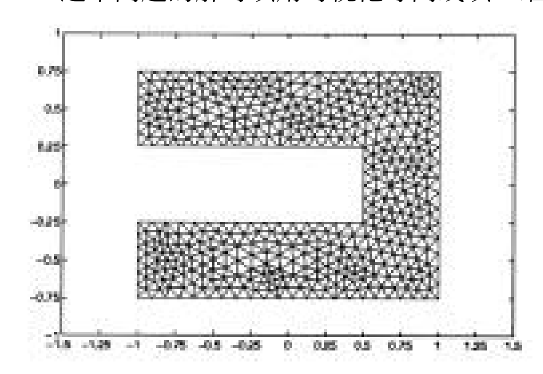


图1-11　PDE工具箱采用FEM的三角形子区域

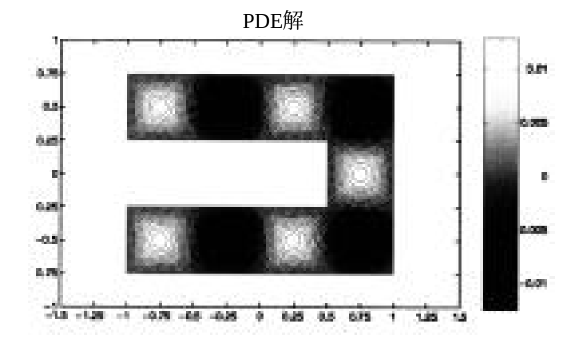


图1-12　PDE工具箱计算的等高线解的图示

■

■ 例1.9　MATLAB中的编程

MATLAB是可编程的。命令序列可以在文本编辑器上写入，然后可以调用 MATLAB命令窗口上的用户定义函数或程序。文件的名字必须要有扩展名 **.m**，这个文件就称为 **M文件**。这些M文件可以用相同的方法象标准MATLAB函数一样使用。

(a) 阶乘n！=1×2×3×⋯×n可以用不同的方法计算。这里，展示一个递归的用户定义函数：

```
function p =factorial(nn)
%计算nn的阶乘。
if  nn == 0
  p = 1;
else
  p = nn*factorial(nn-1);
end
```

这个M文件名叫**factorial.m**，对这个函数的调用如下所示：

```
fourfactorial=factorial(4)
```

其结果为：

```
fourfactorial=
    24
```

(b) 在MATLAB中可以使用面向对象的程序设计。有一个类 **world**，用来模拟一个小世界内的生与死。对于一个要在这个世界出生的个人来说，他需要一些邻居，但太多的邻居会构成生存危机。类**world**的命令是在目录@world中的文件**world.m**中。

```
Function w=world(Size, Density, nrCreate, nrSurvive)
% WORLD 生成一个生命世界。
% 这是命令方式。
% Size给出这个世界的边，它是方的。
% Density应该是在0与1之间，定义了实与空的近似比率。
% nrCreate一个需要生长的空单元的最近邻居的号码。
% nrSurvive一个需要生存和长大的满单元的最近邻居的号码。

colormap('gray')
w.Size = Size;
w.nrCreate = nrCreate;
w.nrSurvive = nrSurvive;
w.Map = round( (-0.5+Density)+rand(Size,Size) );
w.NextY = zeros(Size,Size);
w = class(w,'world');
```

属于类的这种方法在目录 @**world**中定义，但不在这里说明。现在，人们可以通过下列命令产生一个对象**w**：

```
w=world(20, 0.6, [2 4 5], [3 6 7]);
```

现在重复

```
w=year(w)
```

与人类所经历的10年一样，反复10次，得到图1-13。

关于MATLAB中的有关编程信息可参见第12章。

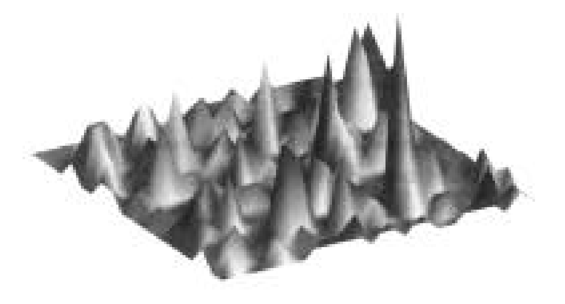


图1-13 10年后世界w上的人口

■

■ 例1.10 图形用户界面

MATLAB提供了一个设计易于使用的程序的机会。程序能够在一个控制图上由按钮、弹出菜单、可编辑文本等控制，这部分在14.3节中描述。在14.4节中，演示了一个由瑞典科学家

设计的图形用户界面的实例。这个程序用于解决非等距网格的一个模型方程。

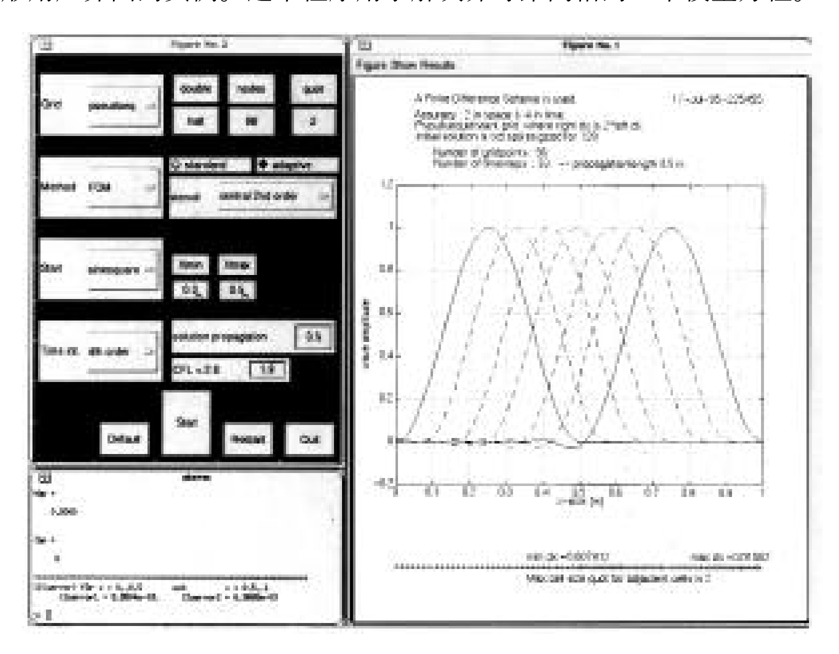


图1-14　带图形用户界面应用的一个MATLAB程序

1.3　MATLAB帮助

本书打算展示MATLAB中可用的命令，并解释如何使用 MATLAB。对于读者，多数时候这已足够，但对想学习更多内容的读者来说，在 MATLAB中可以使用帮助。在系统提示符下，键入help命令，MATLAB给出这个命令的解释。

在MATLAB5 中也包含了一个强有力的HTML格式的帮助数据库。这使得寻找一个命令更容易，依参考而得到帮助。当给出 `helpdesk`命令时，一个浏览器如 Netscape Navigator或Microsoft Internet Explorer就运行，即载入一个索引页。

如果这还不能满足，可向MathWorks公司订购MATLAB的完整手册，具体方法参见前言。



下载

第2章　MATLAB启动

首先描述如何启动和如何退出 MATLAB，这一点很重要。然后描述如何进行分配和计算，也演示如何储存结果、获得帮助和定义你自己的函数。本章的部分内容对 MATLAB老用户可以跳过不读，但是，建议快速浏览本章，也建议参见附录 A“ MATLAB初步 ”。

2.1　启动和退出MATLAB

不同的计算机系统，MATLAB的启动也不一样。在Windows 和Macintosh系统中，程序通常通过点击一个图标而启动。在UNIX系统中，程序是通过在命令行系统提示符后键入如下字符启动：

```
matlab
```

如果上述工作有问题，可请教系统管理员。当启动 MATLAB时，如果 **matlabrc.m**和 **startup.m**文件存在，则执行这些文件。在这些文件中，为满足个人需要，用户可以给定命令以调整MATLAB，例如，constants用于设置图形等。在一个多用户系统上，系统管理员存储 **matlabrc.m**文件，但你也能为自己的使用创建文件 **startup.m**，参见例2.20(c)。

要退出MATLAB，键入quit或exit。

命令集1　退出和中断

exit，quit	结束MATLAB会话。程序完成，如果没有明确保存，则变量中的数据丢失。参见2.8节。
Ctrl-c	中断一个MATLAB任务。例如，当 MATLAB正在计算或打印时，中断一个任务，但会话并没有结束。

除此之外，对一些系统有指定的菜单选择。例如，在 Windows和Macintosh系统中，在文件菜单下可以找到选项quit。

当编辑或执行MATLAB时，下列的快捷键十分有用。通常因为不同的平台使用不同的键，因此，给定了一些替换键。在你的系统上试一下这些键，注意哪些键组合使用。

命令集2　特殊的功能键

↑或 Ctrl_p	恢复前面的命令。
↓或 Ctrl_n	恢复当前命令之后键入的命令。
→或 Ctrl_f	向右移动一个字符。
←或 Ctrl_b	向左移动一个字符。
Delete, Backspace	删除字符。
Ctrl_l 或 Ctrl_←	向左移动一个字。
Ctrl_r 或 Ctrl_→	向右移动一个字。



`Ctrl_a` 或 `Home`	移动到行的第一个字符。
`Ctrl_e`	移动到行尾。
`Ctrl_k`	删除到行尾。
`cedit`	在不同的快捷键间转换。键入help cedit可得到更多的信息。

2.2 MATLAB中矩阵和多维矩阵介绍

MATLAB中数据的基本格式是矩阵。二维矩阵是一个带有以行和列排列的元素的矩形表。如果有m行、n列，这个矩阵的大小就是$m\times n$。多维矩阵的维数大于2，就是说其大小为$m\times n\times\cdots\times p$。

■ **例**2.1

一个2×3的矩阵如下：

$$\mathbf{A}=\begin{pmatrix}1 & 2 & 3\\4 & 5 & 6\end{pmatrix}=\begin{pmatrix}a_{11} & a_{12} & a_{13}\\a_{21} & a_{22} & a_{23}\end{pmatrix}$$

第1行是(1 2 3)，第2列是$\begin{pmatrix}2\\5\end{pmatrix}$。

■

矩阵的元素，即数a_{ij}，通常是实数，但也可以是复数。一个 a_{ij}是指第i行、第j列的数。在例2.1中，有a_{21}=4。在本章，仅涉及数值矩阵，即矩阵包含的仅是数字。包含字符文本的矩阵(见命令集4)和细胞矩阵，在5.5节中介绍，这些矩阵包含了不同类型的数据。

当矩阵仅由一行组成时，它是一个特例，就是一个行向量。如果矩阵仅有一列，就是一个列向量。向量是矩阵的特例。向量中元素的数量是向量的长度。

如果矩阵的维数是1×1，它是一个标量，即是一个数。

在MATLAB中，一个变量可以通过给它分配一个值来定义，如下所示：

```
variable =expression
```

在expression之后按回车键。表达式可以是由数字、变量、操作符和函数等组成。

定义一个变量的另一个方法是输入 expression项，然后MATLAB对预定义变量**ans**(answer的缩写)分配这个表达式值。

二维矩阵的分配可以有多种方法实现。最简单的方法是由方括号 [](参见help paren)包围的逐行给定元素。如果定义一个标量，则方括号就不需要了。

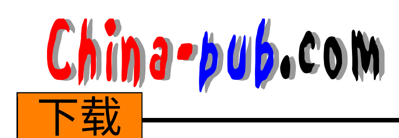
相同行中的元素是由一行或多个空格‘ ’或一个逗号‘，’分隔，列由分号‘；或回车键分隔。没有结尾分号的每个命令在屏幕上显示出其结果。若结尾带分号，就执行计算，但计算结果并不显示。在MATLAB中使用的标点符号的一览表可以通过输入`help punct`得到。

一个变量的值可以通过输入它的名字和按回车键获得， MATLAB以显示这个变量的名字和值作为回答。如果这个变量并不存在，就显示一个错误信息。显示一个变量内容的另一个方法可参见5.1.3节。

一个矩阵或一个向量的指定元素是由指定它的索引来决定。例如：二维矩阵：

```
variable (rowIndex, columnIndex)
```

如果这个变量是一个向量，就只允许有一个索引。如何处理多维矩阵将在例 2.3中讨论。



■ **例**2.2

(a)一个标量的指定。如果写入x=7，则在屏幕上打印如下：

```
x=
   7
```

(b) 如果仅写入7，则结果变为：

```
ans=
     7
```

(c) 2×3维情况下一个矩阵的定义可以通过逐行给出其元素：

```
A=[1  2  3
4  5  6]
```

还在屏幕上给出如下结果：

```
A=
1  2  3
   4  5  6
```

(d) 也可以在同一行上用分号来分隔行以给出所有的元素：

```
A=[ 1  2  3;4  5  6];
```

在命令后的一个分号禁止打印出结果。

(e)一个行向量和一个列向量的定义：

```
rowvec=[1.2  3.2  4];
colvec=[2.7; 3.4;-9.2];
```

(f) 显示一个变量的值。输入colvec，MATLAB显示：

```
colvec=
    2.7000
    3.4000
   -9.2000
```

(g) 逐个元素地分配矩阵：

```
B(1, 1)=1;
B(1, 2)=7;
B(2, 1)=-5;
B(2, 2)=0
```

得到的结果是：

```
B=
    1   7
   -5   0
```

■

三维矩阵和其他数据结构在函数自变量中依次使用行、列和页维数次序。对于多维矩阵有两个索引原理，最自然的就是**矩阵索引**，它给出了每个元素在每一维中的一个位置，例如，在上例中的行和列的索引。图2-1给出了可视化的3×3×2的三维矩阵的不同元素的索引。

另一个索引原理是**线性索引方法**。一些命令把整个矩阵说明为一个长列元素，例子之一是reshape(详见命令集37)。如果所有的元素被放在一行上，那么对各元素给定的一个线性索引号说明了一个指定矩阵的索引给出的某个位置，如图 2-2所示。

创建一个多维矩阵有多种方法。

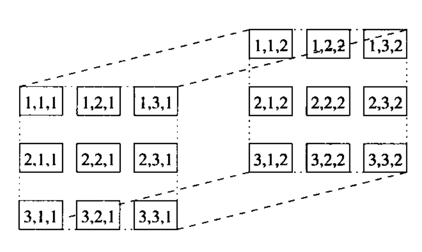


图2-1 一个3×3×2矩阵的元素索引

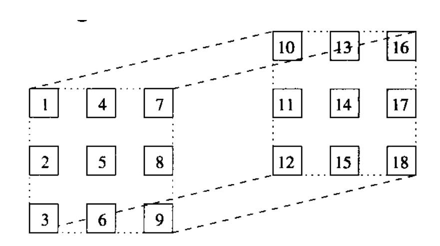


图2-2 一个3×3×2矩阵的线性索引

■ **例**2.3

(a) 有两个二维矩阵**A**和**B**

```
A=
         1     2     3
         4     5     6

B=
        11    12    13
        14    15    16
```

可以很容易地构造一个三维矩阵**C**：

```
C( :, :, 1)=A；
C( :, :, 2)=B；
```

给出：

```
C(:,:,1)=
         1     2     3
         4     5     6
C(:,:,2)=
        11    12    13
        14    15    16
```

(b) 为改变**C**中的一个元素，可以输入：

```
C(1, 1, 1)=100；
```

得到：

```
C(:,:,1)=
       100     2     3
         4     5     6
```

■

MATLAB允许在同一命令行上定义多个变量，也可以在按回车键之前通过输入三个点"……"以便在下一行继续输入。

■ **例**2.4

(a) 在一行上的几个命令：

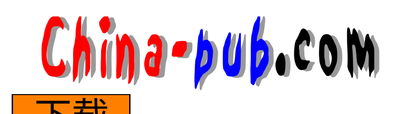
```
x=7；y=4.6735567; z=x^y;
```

(b) 一个长的命令可以分成几行书写：

```
mat1 = [1.2 1.1 -1.1 1.4 1.1 -1.1 -1.2 ...
-1.1 -1.3 1.7];
```

■

MATLAB记忆不同矩阵变量的维数。为了获得一个变量的维数，可以使用命令size和length。



下载

现在，令**A**是一个$m \times n \times \cdots \times p$矩阵，**x**是$m \times 1$矩阵(一个列向量)或是$1 \times n$矩阵(一个行向量)，MATLAB有如下的命令集：

命令集3 变量大小

size(A)	给出包含**A**的维数的一个行向量。在这个返回向量中的第一个元素是行数，随后是列数、页数等。
[m, n, …, p]= size(A)	给出**A**的维数、*m*行数和*n*列数，即两个标量。如果给出的自变量少，则后边的维数加入最后一个自变量。
size(A, dim)	在数组*dim*中给出**A**的维数。
size(x)	给出向量**x**的大小或长度的一个行向量。如果*x*是一个列向量，则第1个元素是*m*，第2个元素是1。如果x是一个行向量，则第1个元素是1，第2个元素是*n*。
length(x)	给出一个向量的长度，即如果 **x**是一个行向量，那么这个向量的长度就是 *n*；如果**x**是一个列向量，那么这个向量的长度就是*m*。
length(A)	给出*m*，*n*，…，*p*的最大数。
ndims	返回多维矩阵**A**的维数。这个函数等价于length(size(A))。
sub2ind(size, m, n, …)	给出维数为**size**的一个矩阵的线性索引号(*m*，*n*，…)。
[m, n, …]= ind2sub (size, ind)	用线性索引**ind**对元素给出索引(*m*，*n*，…)，要把图像矩阵说明为多维，其参数 **size**必须是一个向量。

注意 有时用字dimemsion代替维数和长度，通常它可以表示与一个向量范数相关的长度和与一个矩阵范数相关的矩阵的维数。可是，在本书中，维数和长度的使用与size和length有相同的含义。当这个长度概念以其他意义使用时，就要说明。

■ **例**2.5

(a) 命令thesize1=size(A)，式中的**A**与例2.2(c) 中的相同，结果为：

```
thesize1=
        2    3
```

命令thesize2=size(C)，式中的**C**是例2.3中的三维矩阵，得到：

```
thesize2=
        2    3    3
```

(b)检查图2-2中的信息是否正确，可以使用：

```
sub2ind([3 3 2],1,2,2)

ans =
    13
```

■



2.3 MATLAB中的变量

在MATLAB中，变量名可以有19个字符。字母A～Z、a～z、数字和下划线‘ _ ’可以作为变量名，但第一个字符必须是一个字母。预定义函数名也可以像一个变量名那样使用，但函数只有在变量由命令clear删除后才能使用，所以，不主张这样使用。

MATLAB是区分大小字母的，如矩阵a和A是不一样的。MATLAB命令通常是用小写字母书写。例如，命令abs(A)给出了A的绝对值，但ABS(A)会导致在屏幕上显示如下错误信息：

```
??? Undefined variable or function ABS; Caps Lock may be on
```

在变量使用之前，用户不需要指定一个变量的数据类型，也不必声明变量。 MATLAB有许多不同的数据类型，这对决定变量的大小和形式是有价值的，特别适合于混合数据类型、矩阵、细胞矩阵、结构和对象。

对于每一种数据类型，有一个名字相同的、可以把变量转换到那种类型的函数。所用的不同的基本数据类型如下所示。

命令集4 数据类型和转换函数

double	是一个双精度浮点数，每个存储的双精度数用64位。
char	用于存储字符，每个存储的字符用16位。
sparse	用于存储稀疏矩阵，由一个sparse使用的内存是4+(非零元素数*16)。
unit8	是一个无符号的8位整型数。数学函数并不对使用到的这种数据类型进行定义，如存储图像。

混合数据类型在第5章和第12章中描述。

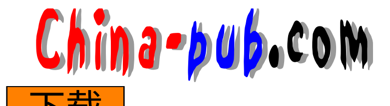
在MATLAB中，有许多功能可以帮助找出一个变量是否是一个特殊类型。也有一个特殊的逻辑向量，它是由命令repmat生成(见第4.1节)。

命令集5 逻辑函数

iscell(x)	如果x是一个细胞矩阵，返回1；否则为0。可参见第5.5节。
isfield(x)	如果x在一个结构中是一个域，返回1；否则为0。可参见第12.5节。
isfinite(x)	返回一个与x相同大小的向量，这个x包含有限元的位置为1，其他位置为0。
islogical(x)	如果x是一个逻辑向量，返回1；否则为0。
isnumeric(x)	如果x是一个数值向量，返回1；否则为0。
isstr(x)	如果x是一个字符串，返回1；否则为0。可参见第5.1节。
isstruct(x)	如果x是一个结构，返回1；否则为0。可参见第12.5节。
isobject(x)	如果x是一个对象，返回1；否则为0。可参见第12.6节。
logical(x)	返回一个可以使用的逻辑向量，例如逻辑索引或逻辑测试。

■ **例**2.6

如果一个函数是要应用到一个矩阵中的隔一个元素上，那么可以这样做：




```
data = rand(1,10)

data =
        0.6700    0.2009    0.2731    0.6262    0.5369    0.0595
        0.0890    0.2713    0.4091    0.4740
```

使用repmat创建x：

```
x = repmat([1 0],1,5)

x =
      1     0     1     0     1     0     1     0     1     0

filter = logical(x)

filter =
      1     0     1     0     1     0     1     0     1     0
```

为使用滤波器，给出下面的命令：

```
halfdata = data(filter)

halfdata =
        0.6700  0.2731  0.5369  0.0890  0.4091
```

为使用与函数round一起的过滤器，可输入：

```
result = round(data(filter))

result =
      1     0     1     0     0
```

命令repmat产生块矩阵，第 4.1 节将进一步对此进行描述。 ■

在MATLAB中有许多如下的预定义变量：

命令集6　MATLAB中预定义变量

ans	分配最新计算表达式的值，这个表达式并没有给定一个名字。
eps	返回机器精度，定义 1与最接近可代表的浮点数之间的差。 *eps*数在一些命令中用作偏差。用户可以设定一个新的 *eps*值，但要注意这个*eps*值不能由命令clear恢复。
realmax	返回计算机能处理的最大浮点数。
realmin	返回计算机能处理的最小的非零浮点数。
pi	返回π，即3.141592653589793，如果*eps*足够小，那么用16位十进制数来表示其精度。
inf	定义为1/0。当出现被零除时， MATLAB就返回*inf*，并不中断执行而继续计算。
NaN	定义为“Not a Number”，这个非数值要么是%类型，要么是inf/inf。



`i, j`	定义为 $\sqrt{-1}$，虚数单位。可以为 i和j分配其他值，它们将不再是预定义常数。可以由clear命令恢复。
`nargin`	给出在一个函数调用中输入自变量的个数，可参见第 12.3节。
`nargout`	给出在一个函数调用中输出自变量的个数，可参见第 12.3节。

为寻找哪个变量可以被定义，可以使用下列命令集：

命令集7　变量列表

`who`	列出已定义的变量。
`Who global`	与who相同，但仅列出全局变量。参见第 12.3节。
`who a*`	给出所有以a开头的变量的一个列表。
`whos`	给出比命令who更详细的列表，如显示矩阵的维数。
`whos global`	与whos相同，但仅列出全局变量，可参见第 12.3节。
`exist(namestr)`	根据在字符串 **namestr** 中的变量的定义，返回不同的值。关于字符串更多的信息可参见第 5章。这里要注意的重要的一点是变量名应在引号‘ ’之间给出。函数返回值的情况是： 1) 表示**namestr**是一个变量名； 2) 表示**namestr**是一个M文件名(参见第2.9 节)； 3) 表示**namestr**是一个MEX文件名(参见第15章)； 4) 表示**namestr**是一个编译的SIMULINK函数； 5) 表示**namestr**是一个预定义的MATLAB函数名。
`inmem`	返回一个带字符串的细胞向量，这个字符串包含目前在内存中的函数、M文件。如果给出两个输出参数，则第二个包含了一个目前在内存中的MEX文件的列表。
`workspace`	对由whos得到的信息给出一个图形界面。命令clear被集成在这个环境中。由help workspace 给出的信息，也在UNIX下工作。

如果不被用户删除或重命名，每个被定义的变量将在整个过程中保留。要删除变量，系统会劝告如要处理大矩阵可用命令 clear。

命令集8　删除变量和合并

`clear`	删除所有变量并恢复除 *eps*外的所有预定义变量。可参见命令集111当运行文件时的clear。
`clear name`	仅删除变量 *name*。
`clear name1 name2 …`	删除变量*name*1、*name*2、…
`clear a*`	删除所有 *a*开头的变量。
`clear value`	根据*value*给出不同的结果。键入help clear 可得到更多的细节。



下载

`pack`	重组和压缩已分配的内存碎块。当MATLAB的内存满后，可以使用命令pack而不是清除任何变量来得到更多的空间。将会产生如下情况： • 所有变量都会保存在磁盘上的一个临时文件 **pack.tmp**中； • 删除主内存中的内容； • 所有变量将从**pack.tmp**加载到主内存中； • 删除文件**pack.tmp**。
`pack filename`	用文件**filename**作临时文件，重组和压缩已分配内存。

键入help clear、help memory和help pack可以获得更多的有关如何在MATLAB节省内存的信息。

注意　MATLAB中的命令实际上可以看作函数，把字符串看作自变量，这就意味着有两种描述：

```
command argument
```

```
command('argument')
```

这两种描述是等价的。例如，`clear name`与`clear('name')`得到相同的结果，`who global`与`who('global')`等价。其他的例题可在本书的其他几个地方找到，例如`axis spuare`与`axis('square')`在第13.3节中可找到。由于命令自变量可以通过使用字符串控制命令而产生，使用函数/字符串公式的可能性使得MATLAB编程语言十分灵活，参见第5章。

2.4　算术表达式和数学函数

在MATLAB中通常的惯例是书写数字。对十进制数，使用科学记数法可以书写十分大和十分小的数。例如3.14和1.23E－6，这里，后者代表1.23×10^{-6}。

MATLAB有算术运算符的扩展集，它们是：

1) ^　　幂
2) *　　乘
　 /　　右除(正常除)
　 \　　左除
3) +　　加
　 －　　减

这是按序给出的运算，1是最高优先级。在带相同优先级的运算符表达式中，按从左到右的顺序执行。圆括号()能够用于改变优先级次序。

在第3.3节中，将看到两种不同的除法是有用的。对于数量右除2/5得0.4与左除5\2是相同的，斜线号“靠着”的表达式或数字是分母。

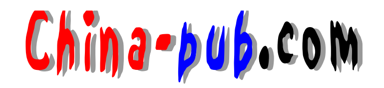
■ **例2.7**

如果书写`a/b+c`，MATLAB读作 $\frac{a}{b}+c$ ，但表达式`a/(b+c)`被读作 $\frac{a}{b+c}$ 。

如果使用左除`a\(b+c)`，MATLAB把它译成 $\frac{b+c}{a}$ 。

■

MATLAB包含了预定义数学函数，它们可以用于算术表达式中。如果自变量是复数，那么，



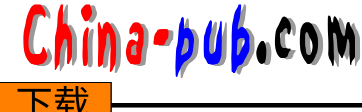
多数情况下是答案。

MATLAB也能计算含有预定义变量的表达式：例如，一个表达式可以用作一个函数的自变量。

预定义数学函数在命令集9中列出。即使这些函数是为数量自变量描述的，将在第 3.6节看到它们既能处理向量，也能处理矩阵。要注意的是所有三角函数都需要其自变量以弧度表示。

命令集9　数学函数

命令	说明
`abs(x)`	求x的绝对值，即 $\lvert x\rvert$。
`sign(x)`	求x的符号，如果是正的得1；负的得－1；零得0。
`sqrt(x)`	求x的平方根，即$\sqrt{x}$ 。
`pow2(x, f)`	求$x\times 2^f$。把 f加到x的浮点格式下的指数上计算是一种十分有效的运算。
`exp(x)`	求x的指数函数，即e^x。
`log(x)`	求x的自然对数，即$\ln x$。
`log10(x)`	求x以10为底的对数，即$\log_{10}x$。
`log2(x)`	求x以2为底的对数，即$\log_2 x$。
`sin(x)`	求正弦x，x为弧度。
`cos(x)`	求余弦x，x为弧度。
`tan(x)`	求正切x，x为弧度。
`cot(x)`	求余切x，即$1/(\tan x)$，x为弧度。
`asin(x)`	求反正弦，即$\sin^{-1}x$。
`acos(x)`	求反余弦，即$\cos^{-1}x$。
`atan(x)`	求反正切，即$\tan^{-1}x$。
`atan2(x,y)`	求四象限反正切(x/y)，其结果在$[-\pi, \pi]$区间内。
`acot(x)`	求反余切x=四象限反正切$(1/x)$。
`sec(x)`	求正割x，即$1/(\cos x)$。
`csc(x)`	求余割x，即$1/(\sin x)$。
`asec(x)`	求$\sec^{-1}x=\arccos(1/x)$。
`acsc(x)`	求$\csc^{-1}x=\arcsin(1/x)$。
`sinh(x)`	求双曲正弦x。
`cosh(x)`	求双曲余弦x。
`tanh(x)`	求双曲正切x
`coth(x)`	求双曲余切x，即$1/(\tanh x)$。
`asinh(x)`	求$\sinh^{-1}x=\ln(x+\sqrt{1+x^2})$ 。
`acosh(x)`	求$\cosh^{-1}x=\ln(x+\sqrt{1-x^2})$ 。
`atanh(x)`	求$\tanh^{-1}x=0.5\ln((1+x)/(1-x))$。
`acoth(x)`	求$\coth^{-1}x=0.5\ln((x+1)/(x-1))$。
`sech(x)`	求双曲正割x，即$1/(\cosh x)$。
`csch(x)`	求双曲余割x，即$1/(\sinh x)$。
`asech(x)`	求$\mathrm{sech}^{-1}x=\ln((1+\sqrt{1-x^2})/x)$ 。
`acsch(x)`	求$\mathrm{csch}^{-1}x=\ln((1+\sqrt{1+x^2})/x)$ 。



■ **例**2.8

(a) 如果键入sinepi=sin(pi)，就得到：

```
sinepi=
     1.22466e−16
```

这个结果并不是精确地为0，因为pi是π的近似值，在计算中有舍入误差。

(b) logarithm = log10(100)

```
logarithm =
     2
```

(c) e = exp(1)

```
e =
     2.7183
```

■

在MATLAB中有几个命令用于数的取整。在命令集 10中，*x*是一个浮点数或是一个带浮点元素的矩阵。

命令集10　取整命令和有关命令

round(x)	求最接近*x*的整数。如果x是一个向量，则适用于所有元素。
fix(x)	求0方向最接近x的整数。即负x向上四舍五入，正x向下四舍五入。
floor(x)	求小于或等于*x*的最接近的整数。
ceil(x)	求大于或等于*x*的最接近的整数。
rem(x，y)	求整除*x*/*y*的余数。
gcd(x，y)	求整数*x*和*y*的最大公因子。
[g，c，d]=gcd(x，y)	求*g*，*c*，*d*，满足*g*=*xc*+*yd*。
lcm(x，y)	求正整数*x*和*y*的最小公倍数，也能用于决定最小公因子。
[t，n]=rat(x)	由有理数*t*/*n*求*x*的近似值，这里的*t*和*n*是整数，相对误差小于10^{-6}。也可参见rats(第5.1.2节)，它给出了对应的字符串。
[t，n]=rat(x，tol)	与上相同，但相对误差小于*tol*。
rat(x)	求*x*的连续的分数表达式。
rat(x，tol)	求带相对误差*tol*的*x*的连续的分数表达式。

■ **例**2.9

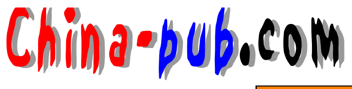
(a) 取整有几种方法，命令是：

```
x = -1.49;
rdx = round(x), fixx = fix(x), flx = floor(x), clx = ceil(x)

return

rdx =
    -1

fixx =
    -1
```




```
flx =
    -2

clx =
    -1
```

(b)由有理数t/n求 $\sqrt{2}$ 的近似值：

```
[t,n] = rat(sqrt(2))

t =
        1393

n =
   985
```

为了与真值比较，键入differ=sqrt(2)－t/n，得：

```
differ=
  3.6440e-07
```

由此可见，差值不大。如果在函数rat对参数tol指定一个较小的值，那么这个差值将会是较小的。

■

在MATLAB中，大多数情况下是允许复数值表达的。加入变量i和j返回**虚数单位**，即$\sqrt{-1}$的值，能用于产生复数。这是可以用名字i和j作为变量的名字，一个新的复数单位可以由此产生：

```
ii=sqrt(－1)；
```

由于空格是分隔元素的，因此在书写复数元素时要慎用空格。参见下面例题 2.10(c)。

■ **例**2.10

(a) `z = 3 + 4i`

```
z =
    3.0000 + 4.0000i
```

(b)一个较复杂的表达式：

```
w=r*exp(i*theta)；comp=z*w；
```

式中，r和theta是一个已经定义的变量。

(c) 向量也可以是复数；

```
complexvector=[1－i  2－2i  3  －3i ]
```

返回：

```
complexvector=
 1.0000－1.0000i    2.0000－2.0000i    3.0000  －3.0000i
```

注意，在3与－3i之间的空格使MATLAB读取它们时看作为两个分隔的复数。如果键入下列内容，可以看到此结果：

```
length(complexvector)
```

给出的结果是：

```
ans=
    4
```

■



MATLAB中有一些处理复数和函数的命令。

命令集11 有关复数的函数

`real(z)`	求z的实部。
`imag(z)`	求z的虚部。
`abs(z)`	求z的绝对值，即 $\|z\|$。
`conj(z)`	求z的复数共扼，即$\bar{z}$。
`angle(z)`	求z的相角，即$z=x+iy=re^{i\theta}$中的θ。
`unwrap(v)`	求与**v**相同长度的向量。这里，两个相邻元素间的相角差已经改变，因此，差最大为π
`unwrap(v, k)`	求出如上的一个向量，但用转移偏差k代替π
`cplxpair(v)`	给出一个**v**中各元素按实部递增排序，并使其复数组合成复数共扼对的一个向量。在一个共扼对中，负虚部在前，实元素排在向量的后部。如果**v**的一个元素在**v**中没有它自己的复数共扼，则显示一个错误信息。

■ **例**2.11

令复数z为：

```
z=1+2i；
```

(a) z的实部和虚部由下面求出：

```
realpart = real(z), imagpart = imag(z)

realpart =
     1

imagpart =
     2
```

(b) 复数共扼由`conjugate=conj(z)`求出：

```
conjugate=
   1.0000－2.0000i
```

z的绝对值由`absz=abs(z)`求出：

```
absz=
    2.2361
```

(c) 一个复数的自变量，即复平面中的相角由`arg=angle(z)`求出：

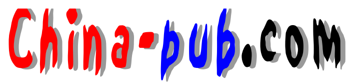
```
arg=
    1.1071
```

■

MATLAB也用于坐标系之间转换的函数。这些函数既能对向量、也能对矩阵进行运算，其结果将有与输入自变量相同的维数。

命令集12 坐标转换

`[theta, r]= cart2pol(x, y)`	将笛卡尔坐标转换为极坐标，极坐标*theta*和*r*是由卡笛尔*x*和*y*得到。

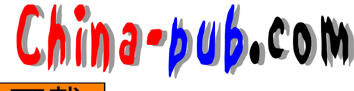


[x，y]=pol2cart(theta，r)	将极坐标转换为笛卡尔坐标，笛卡尔坐标x和y是从极坐标$theta$和r得到。
[alpha，theta，r]=cart2sph(x，y，z)	将卡笛尔坐标转换为球坐标，角$alpha$、$theta$和长度r是从笛卡尔坐标x、y和z得到。
[x，y，z]=sph2cart(alpha,theta,r)	将球坐标转换为笛卡尔坐标x、y和z。

在MATLAB中也有更高级的预定义数学函数。

命令集13 特殊的数学函数

legendre(n，x)	返回一个n+1长度的向量，代表与n次legendre函数相关的值和对x的0到n次计算的值。如果x=**x**是一个向量，则这个命令返回一个矩阵，列是为**x**的每一个元素计算的Legendre函数值。**x**的每个元素都必须在[−1，1]范围内。
bessel(n，x)	求第1类贝塞尔函数，n和x可以是向量，但n必须是步长为1的递增，范围为[0，1000]。这个命令根据**x**是否是复型而调用不同的程序，但这些程序能被直接调用。输入help bessel可以得到更多的信息。
besselу(n，x)	求与bessel有相同自变量的第2类贝塞尔函数。
gamma(x)	求γ函数，即对正x： $\Gamma(x)=\int_0^{\infty} t^{x-1}e^{-t}\,dt$ 要获得负x的定义，键入help gamma。
gammainc(x，a)	求不完全γ函数 $\frac{1}{\Gamma(a)}\int_0^{x} t^{a-1}e^{-t}\,dt$
gammaln(x)	求γ函数的自然对数。用log(gamma(x))可以避免上下溢出。
beta(x，y)	求ß函数，即： $\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$ 自变量x必须在[0，1]间隔内，如果由三个自变量调用此函数，可以使用下面的命令betainc。
betainc(x，a，b)	求不完全β函数，类似地定义不完全γ函数。
betaln(x，y)	求β函数的自然对数。
expint(x)	求 $\int_x^{\infty}\frac{e^{-t}}{t}\,dt$



erf(x)	求误差函数，即积分：$\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$
erfinv(y)	求逆误差函数。
erfc(x)	求互补误差函数1－erf(x)。
erfcx(x)	求成比例的互补误差函数。键入 help erfcx可得到更多信息。
[k, e]=ellipke(m)	求第1和第2类0<*m*<1的完全椭圆函数
[j1, j2, j3]=ellipj(x, m)	求Jacobi椭圆函数。

此外，用户可以定义自己的函数，参见第 2.9节。一些特殊的数学函数在第 10.4节中叙述。

2.5 计算浮点运算次数和时间管理

当相互比较不同算法时，计算一个会话期或一个部分会话期中的算术运算的次数是十分有用的。为了获得浮点运算 (flop)的近似次数，使用命令flops。显示部分MATLAB程序所占用时间的计时方法将在第 12.7节中描述。

命令集14 浮点运算计数器

flops	返回完成浮点运算的近似次数。最新的计数器值是 0，这是MATLAB启动时的缺省值。 加法和减法运算时，如果是实数运算，计入1个运算；如果是复数运算，计入2个运算。乘法和除法时，如果是实数运算，计入 1个运算；如果是复数运算，计入6个运算。调用初等函数时，如果自变量是实数，计入1个运算；如果是复数，计数较多，具体次数随具体的函数而定。
flops(0)	计数器重置为零。

■ **例**2.12

计算运算的次数。命令如下：

```
flops (0); x = 10 + 20 + 30*40/50;
numflops = flops
```

如期望的一样，其结果为：

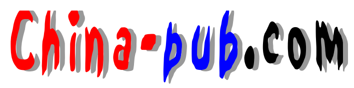
```
numflops=
     4
```

■

MATLAB能够告诉你日期和时间，并且给出计算机的有关信息。与命令 flops一起使用，这些命令可用于分析一个算法的有效性。

命令集15 时间和日期(一)

tic	启动一个可用命令toc读的时钟。
toc	读时钟，即显示开启时钟以来的时间。如果时钟没有运行,toc返回0值。



`clock`	返回用十进制数表示日期和时间的具有 6个元素的行向量。前5个元素是整数，秒由几个十进制小数表示。命令 `fix(clock)`四舍五入至最接近整数的秒。
`etime(t1, t2)`	计算**t1**和**t2**时间间隔内所消耗的时间，以秒计算。 **t1**和**t2**是表示日期和时间的6个元素的行向量。
`cputime`	返回以秒计的MATLAB自启动以来所用的CPU时间。

■ **例**2.13

下列方法可进行时间操作：

写入下式保存当前时间

```
t1=clock
```

写入MATLAB命令

```
timedifference=etime(t1, clock)
```

得到完成上述两条命令所用的间隔时间。

■

MATLAB有处理日期的内部函数。一些系列函数的使用组成了一个描述日期的整数部分和一个描述时间的小数部分。

命令集16 时间和日期(二)

`date`	以日—月—年字符串形式返回当前日期。
`calendar(yyyy, mm)`	显示当年*yyyy*当月*mm*按周排列的6×7矩阵形式的日历。
`datenum(yyyy, mm, dd)`	给出当年*yyyy*当月*mm*当日*dd*的序列数字。日期0000—01—01为1天。
`datestr(d, form)`	返回*form*格式的序列数字表示的日期，见表2-1。
`datetick(axis, form)`	用于在图中的坐标轴上写数据。
`datevec(d)`	如果*d*是一个由诸如`datestr`返回的格式表示的序列数字或日期，则返回一个向量 [*yyyy mm dd ho mi se*]。
`eomday(yyyy, mm)`	返回当年当月的天数。
`now`	返回当天和当时的序列数字。
`[daynr dayname]= weekday(day)`	返回*dayname*以告知当天*day*是否星期天、星期一等。 *daynr*等于一周中当天 *day*的序列数字。这里的 *day*是一个字符形式或序列数字的日期。

在`datestr`中可以使用的不同的日期格式如表 2-1所示。用1986年4月26日2时14分作为例

子列表介绍。

表2-1 日期格式

0	格式*dd mmm yyyy HH:MM:SS*	*26—Apr—1986 02:14:00*
1	格式*dd mmm yyyy*	*26—Apr—1986*
2	格式*mm/dd/yy*	*04/26/86*
3	格式*mmm*	*Apr*
4	格式*m*	*A*
5	格式*m#*	*4*
6	格式*mm/dd*	*04/26*
7	格式*dd*	*26*
8	格式*ddd*	*Sat*
9	格式*d*	*S*
10	格式*yyyy*	*1986*
11	格式*yy*	*86*
12	格式*mmmyy*	*Apr86*
13	格式*HH:MM:SS*	*02:14:00*
14	格式*HH:MM:SS PM*	*2:14:00 AM*
15	格式*HH:MM*	*02:14*
16	格式*HH:MM PM*	*2:14 AM*
17	格式*QQ-YY*，这里*QQ*表示几刻钟	*Q2—86*
18	格式*QQ*	*Q2*

■ **例**2.14

为了获取指定日期是星期几，使用如下语句并得到相应结果：

```
[dnr dname] = weekday('26--April--1986')

dnr =
          7

dname =
          Sat
```

■

在MATLAB中，一周是从星期天开始的，这使得星期六的星期数为 7。

2.6 输出格式

在屏幕上，通常以不带小数的整数格式或带四位小数的短浮点数格式显示结果。

如果一个矩阵中所有的元素都是整数，那么它们将以整数格式显示。但是，如果有一个或一个以上的元素是非整数，则所有的元素都以浮点数格式显示。

输出格式在计算中不影响精度， MATLAB总是按高精度完成计算。对大多数的计算机而言，MATLAB在计算中使用16位小数。

命令`format`用于改变输出格式。在 Windows和Macintosh版本中，输出格式也能通过命令窗体中的下拉菜单控制。



命令集17 数字输出格式

`format defformat`	将输出格式改为由*defformat*定义的格式，这类格式可以是如下之一：short、long、short e、long e、hex、+、bank、rot。也有compact或loose，它给出了一个较紧缩或较宽松的输出格式，但并不影响数值输出格式。本书中的所有例题都使用format compact。
`more on`	当全屏时停止显示，在进一步删除显示之前等待键盘输入。在窗体底部，MATLAB打印—*more*—以指示有更多的信息要显示。
`more off`	不考虑窗体是否足够大而给以输出。
`more(n)`	如果输出多于*n*行，则显示输出*n*行。

■ **例2.15**

设 p=1+1/3，先定义格式，然后在屏幕上显示 p：

format short	得 1.3333	4位小数
format long	得 1.33333333333333	14位小数
format short e	得 1.3333e+00	4位小数
format long e	得 1.333333333333333e+00	15位小数
format hex	得 3ff5555555555555	16进制数
format +	得 +	正：+ 负：－或零：0
format bank	得 1.33	美元和美分
format rat	得 4/3	作为一个有理数

■

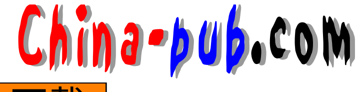
MATLAB在输出中使用的空格数是可以减少的。当输出行长于窗体时。也可以引导输出。本书在实例中使用了format compact来减少空格数。

2.7 帮助命令和示范

帮助总是可以通过使用下列命令中的任何一条获得。

命令集18 帮助命令

`help`	给出大约20个主题的列表，每个主题给出了基本信息。这些主题以目录给出，有关每个主题的信息通过help dir给出，这里的dir是目录。
`help command`	对指定的命令给出帮助。
`help dir`	给出目录dir的内容。
`hthelp`	对一个超链接数据库打开一个 MATLAB GUI。这个命令用helpdesk替换，也可能在今后的版本中删除。
`htpp`	是链接到hthelp中的帮助文件的一个预处理器。
`loadhtml`	通过hthelp和http用于加载、中断和显示HTML文件。



`helpdesk`	在MATLAB帮助桌面上启动一个带索引页的Web浏览器。
`doc command`	在MATLAB帮助桌面上得到命令*command*的帮助页。
`web URL`	将浏览器指向*URL*，如果需要则打开一个浏览器。参见 `help web`可以得到更多信息。
`lookfor text`	在所有的M文件的第1行中查找字符串**text**。
`demo`	给出一个MATLAB的不同命令、函数和应用领域的示范。命令demo运行MATLAB Expo，它显示选择不同示范实例的一个菜单。也可以找到几个简单的游戏。
`expo`	运行MATLAB Expo；也可参见`demo`。
`info`	给出有关MATLAB的信息。例如，什么种类的计算机能够运行MATLAB，如何得到更多的有关 MATLAB最近的进展和新版的信息等等。
`whatsnew`	给出新版本中新命令的有关信息。
`subscribe`	使之变成一个MATLAB的预约用户。
`why`	解释为什么一些事情会出错。

MATLAB Help Desk是一个基于HTML的帮助系统。在这个系统里，通过 `help`得到的大量信息是以超链接形式一起提供文献、图片和公式。它也含有一个比由 `lookfor`提供的更好的查找工具。重要的是要注意 MATLAB Help Desk使用了Java Script，因此，Java Script必须是使能的。如果给出 `helpdesk`命令，浏览器并不如图 2-3那样，而是仅显示 MATLAB标记，这可能是Jave Script并没有生效。在UNIX 中，可以通过一个名为**docopt.m**的M文件决定浏览器的选择。在Windows中，MATLAB使用与.htm文件相关的程序。

■ **例**2.16

(a) 命令`help size`得到信息：

SIZE 矩阵的维数。
D=*SIZE*(*X*)，对于*M*×*N*的矩阵*X*，返回两个元素的行向量*D*=[*M*，*N*]，其中包含了矩阵中行数和列数。对于*N*-*D*数组，*SIZE*(*X*)返回一个1×*N*数组长度的向量。
[*M*，*N*]=*SIZE*(*X*)返回输出变量中的行数和列数。[*M1*，*M2*，*M3*，…，*MN*]=*SIZE*(*X*)返回*X*中头*N*个数组的长度。
*M=SIZE(X，DIM)*返回由标量*DIM*指定的数组长度。例如，*SIZE(X，1)*返回行的数量。
也可参见*LENGTH*、*NDIMS*。

重载方法

```
Overloaded methods
   help zpk/size.m
   help tf/size.m
   help ss/size.m
```

注意，尽管MATLAB不接受大写字母给出的命令，但在帮助文本中用大写字母书写命令。

(b) 为启动MATLAB Help Desk用命令`helpdesk`，MATLAB启动如图2-3所示的一个浏览器(或使用一个已经运行的浏览器)。

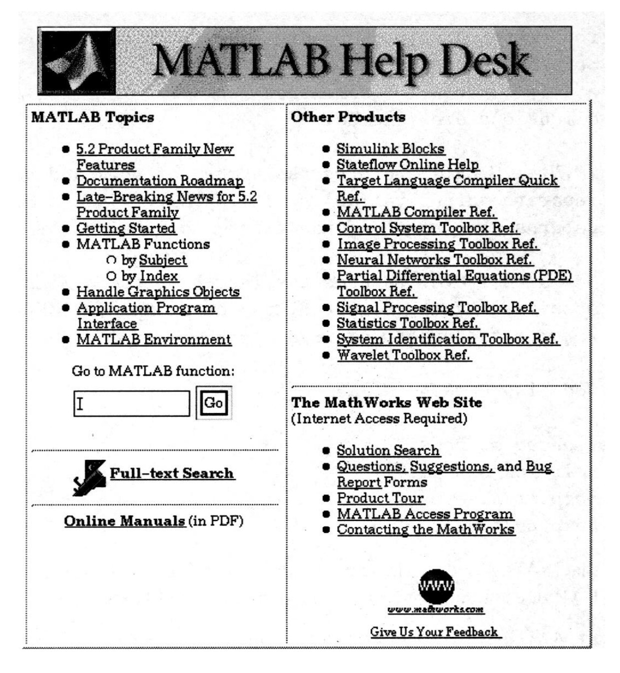


图2-3　启动MATLAB Help Desk后显示的索引页

(c) 查找有关正弦函数信息，可键入：`lookfor sin`，则得到：

ACOS	反余弦
ACOSH	反双曲余弦
ASIN	反正弦
ASINH	反双曲正弦
COS	余弦
COSH	双曲余弦
SIN	正弦
SINH	双曲正弦
TFFUNC	调制的余弦高斯脉冲的时间域和频率域
DST	离散正弦变换
IDST	反离散正弦变换

■

学习MATLAB的一个很好的办法是用命令`demo`运行一个示范程序，然后用命令`help`检查插入的命令，最后开始使用此命令。

MATLAB有许多可以给出有关你正在运行MATLAB的计算机的信息。

命令集19　计算机信息

`[str, n]=computer`	给出正在运行MATLAB的计算机的一个描述。字符串str因计算机或操作系统而定，例如，VAX、Sun、PC和Macintosh等。*n*是当前安装的MATLAB允许一个矩阵中元素的总数量。
`isieee`	对带IEEE算法的计算机，返回1，例如，IBMPC和Macintosh。对不带IEEE算法的计算机，返回0，例如VAX和Cray。

`version`	返回一个带当前MATLAB版本号的字符串。
`ver`	显示当前MATLAB和工具箱的版本号。
`hostid`	返回MATLAB服务器主机识别号。
`getenv(str)`	返回与**str**连接的文本。这里，**str**是一个符号或一个环境变量的名字。
`terminal`	设置MATLAB终端类型。

■ **例**2.17

在Sun工作站上运行Solaris 2，命令为：

```
[comp, numb]=computer
```

返回

```
comp =
    SOL2
numb =
    268435455
```

■

2.8 保存和装载

MATLAB能保持屏幕上显示的日记，这是由命令 `diary`完成的，图形输出则是例外。有关打印或保存图形，可参见第13.7节。

命令集20　会话日记

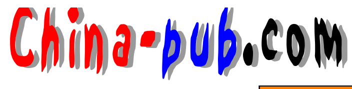
`diary filename`	在文件**filename**中保存下一个会话。
`diary off`	停止记录。
`diary on`	开始记录。继续在当前日记文件中。
`diary`	在文件**diary**中存储下一个会话，但也是在启停日记之间的一个转换器。

产生的ASCII文件以后能被编辑，且包括文档。可是，用命令 `diary`保存的值和结果通常不能由MATLAB在下阶段阅读。

为了保存各种变量和它们的内容以便能在下阶段使用，应该使用命令 `save`和`load`。表中的文件名**filename**确定了MATLAB是如何说明这个文件的。所有以 .mat结尾的文件是二进制文件，所有以其他形式结尾的文件，包括 **filename.**，都是ASCII文件。

命令集21　保存和装入变量

`save`	在文件**matlab.mat**中保存所有的变量
`save filename`	在文件**filename.mat**中保存所有的变量。如果在文件名后以点结束 **filename.**，或加了另外一个后缀，那么MATLAB不能加后缀 **.mat**。
`save filename v1 v2...`	在文件**filename.mat**中保存变量*v*1、*v*2、…等。



`save filename v -ascii`	在文件**filename.mat**中以可读的ASCII格式保存变量*v*的值，写8位小数。
`save filename V -ascii -double`	在文件**filename.mat**中以可读的ASCII格式、带16位小数的双精度保存变量*v*的值。
`load`	从文件**filename.mat**中装入所有变量。
`load filename`	把文件**filename.mat**中所有变量装入MATLAB。如果**filename**没有一个后缀，那么这个文件由save生成。如果此文件有一个后缀，但只有两位，那么应该使用选项-mat。 如果**filename**含有‘.’和后缀，例如**temp.dat**，那么数据就从相应的ASCII文件加载到MATLAB中，作为一个不带‘.’和后缀的名为**filename**的矩阵，此文件可由save file var-ascii 或直接通过使用一个编辑器来产生或作为另一个程序的输出文件。

■ **例**2.18

假设ASCII文件**A.dat**是通过一个编辑器或一个程序创建的，它含有下列数据：

1 4 5

4 2 9

在MATLAB中，可以通过输入下列命令而得到一个矩阵：

```
load A.dat, A
A =
     1        4       5
     4        2       9
```

■

当文件名是以一个字符串保存时，在这个文件上保存数据的例子可参见例5.8(b)，也可参见第15章。在第15章中，讨论了许多高级文件的处理及如何在FORTRAN和C程序中使用文件。

2.9 命令文件和函数文件

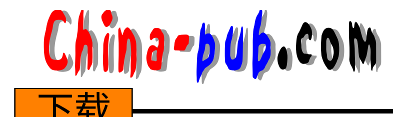
为了代替在MATLAB提示符下输入MATLAB命令的语句，可以把这些命令写入一个文本文件，这个文本文件可用一个编辑器创建。每当用户输入这个文件名和它的自变量时，这些命令就由MATLAB执行。MATLAB从文件而不是从终端读取命令，当文件中最后一个命令被执行时，MATLAB能再从终端读取命令。MATLAB将首先在当前工作目录下寻找此文件，如果它不在当前目录下，那么在该路径下的所有目录中搜索。该路径保持在matlabpath中，可参见命令集22。如果想执行一个没有放在可以自动搜索处的一个文件，你可使用命令`run`，参见第12章。

M 文件是一种文件：

`filename.m`

即，它必须有后缀**.m**。

一个M文件包含许多连续的MATLAB命令，它也可以引用其他的M文件，可以递归，也



就是说可以自己引用自己。

在MATLAB的实用盘“utility-disk”上有大量的预定义M文件，例如，**cond.m**、**demo.m**、**length.m**和**hilb.m**。要了解这些文件的名称，可以使用命令 `what`，列出由用户定义的和在MATLAB目录中存放的M文件。

命令dir可以代替what命令，这个命令属于MATLAB文件命令集。

命令集22 系统命令

命令	说明
`what` *`dirname`*	列出当前目录下所有的 MATLAB文件。如果给定**dirname**，就列出目录**dirname**下的文件。
`dir`	列出一个目录或子目录中的所有文件。这个命令可以用不同的路径名和程序单。
`ls`	以不同的输出格式列出文件。
`pwd`	列出当前的工作目录。
`delete filename`	删除文件**filename**。
`cd`	改变当前目录。
`type filename`	显示文件 **filename**的内容。如果没有指定后缀，MATLAB就读**filename.m**。
`edit file`	打开一个编辑器。如果给定 **file**，那么这个文件就在编辑器中打开。
`copyfile (file1, file2)`	**file1**复制到**file2**。有关错误处理可参见help copyfile 。
`which filename`	显示由**filename**指定的函数的搜索路径。
`path`	显示MATLAB的目录搜索路径。如果给出带自变量的命令，就改变搜索路径。输入 **help path**可以获得更多的信息。
`matlabpath`	当一个新的搜索路径给定时，作为路径工作，但没有错误处理。
`genpath(`*`directory`*`)`	返回一个新的搜索路径，这个路径是由老的和在MATLABROOT/toolbox下所有路径一起组成。如果**directory**给定，那么在**directory**下所有的目录都被增加替换。
`pathsep`	列出分隔标志，这个标志分开了**matlabpath**中的所有目录。
`partialpath`	列出本地搜索路径。
`editpath`	给出一个图形用户界面。在那里，可以从 MATLAB的搜索路径增加和编辑目录。
`addpath(dir1, dir2, …,` *`flag`*`)`	在MATLAB的搜索路径的开头增加目录**dir1**、**dir2**、…。如果字符串 **flag**给定，且是始端，那么目录被加在始端；如果它是末端，则加在末端。
`rmpath dir`	从MATLAB的搜索路径中移去目录dir。
`pathtool`	这是一个修改搜索路径的图形工具。尽管 help

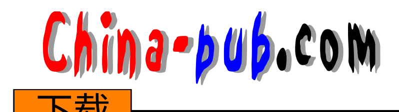


	pathtool建议它不要在 UNIX下工作，但它还是在UNIX下工作。
path2rc	在文件 **pathdef.m**中保存当前的搜索路径，当启动MATLAB时，可以从这个文件中读取搜索路径。
dbtype filename	带行号显示文件 **filename**的内容。如果在 **filename**中没有给定后缀，MATLAB使用后缀.**m**。
dbtype filename r1:rn	带行号地显示**filename**中*r*1行到*rn*行。
lasterr	重复上次的错误信息。
lastwarn	重复上次的警告信息。
!	把这一行看作操作系统的命令，在 Macintosh版本中不能使用。
isstudent	如果使用的是 MATLAB学生版(由Prentice Hall发布的)，返回1；否则返回0。
isdir dirname	如果**dirname**是一个目录，返回1；否则返回0。
isppc	如果在Macintosh Power PC上使用MATLAB，返回1；否则返回0。
isunix	如果在UNIX系统上使用MATLAB，返回1；否则返回0。
isvms	如果在VMS上使用MATLAB，返回1；否则返回0。
vms	从MATLAB中运行一个VMS DCL命令，与！相同，但有可能从这个命令变量中存储数据。输入 help vms可以获得更多的信息。这个命令仅在VMS系统上应用。
dos	从MATLAB中运行一个DOS命令。如同！，见上一条，但有可能从这个命令变量中存储数据。输入 helpdos可以获得更多的信息。这个命令仅在 PC上使用。
applescript	加载一个Apple Script文件并运行它，参见helpdesk了解更多的信息。这个命令仅在 Macintosh上使用。
unix	从MATLAB中执行一个UNIX操作系统命令，如同！。输入help unix可以得到更多的信息。
tempdir	返回一个表示系统中临时目录名的字符串
tempname	返回一个以‘tp’开头的字符串，MATLAB将检查这个字符串是否为系统的临时目录中的一个文件名。因为这样，这个字符串可以用作一个临时文件的名字。
matlabroot	返回带指向MATLAB安装所在目录的搜索路径的一个字符串。

创建一个名为matlab的目录以保存M文件是一个好的主意，MATLAB在这个目录中自动地寻找和发现文件。

■ **例**2.19

(a) 查看包含在MATLAB中的文件**sec.m**的内容，输入：




```
type sec
function y=sec(z)
%  SEC   正割
%   SEC(X)是X的元素的正割。
%   Copyright (c) 1984-98 by The MathWorks, Inc.
%   $Revision: 5.3 $  $Date: 1997/11/21 23:28:33 $

y = 1./cos(z);
```

M文件中的注释，即以%开始的行，在MATLAB中用作文档。这些注释应该是更好地提供信息，而不是象在这个例子中混淆**X**和**z**。基本运算符./在第3.5节中定义。

(b) 如果写入`help sec`，就显示出文件**sec.m**的开始注释行：

```
SEC 正割。
    SEC(X) 是X的元素的正割。
```

■

这是一个函数文件的例子，一个用户定义的函数，即是一个特殊类型的 M文件。除了第二类M文件外，称之为命令文件，一个函数能有一个或几个自变量或参数，它们用分号‘；’分隔。在例题sec中，有一个名为z的参数。所有的参数必须由括号‘()’括起来。命令文件也称为script文件。输入`help script`将得到更多的有关信息。

MATLAB中的函数与C中的函数或FORTRAN的子程序非常相似。函数文件有如下特征：

• 函数文件的第一行必须包含字 function，命令文件没有这种要求。因此，没有这样第一行的M文件是命令文件。

• 第一行必须指定函数名、输入变量(参数)和输出变量(参数)。输入参数是从MATLAB的工作空间复制到函数工作空间的变量。第一行举例如下：

```
function output = name(input)
```

• 一个函数可以有0个、一个或几个输入参数和返回值。

• 与FORTRAN中的子程序、Pascal中的过程或C/C++中的无返回值的函数等价。

```
function = name(inputArguments)
```

在MATLAB中，一个函数M文件可以这样调用：

```
filename(inputArguments)
```

建议函数取名如同文件名一样。调用时所用的变量并不需要与函数文件中定义的变量有相同的名字。

当输入`help name`后，就显示第一行之后的注释。这与命令文件一样使用相似的规定。

函数文件和命令文件的执行如同普通的 MATLAB命令。当输入文件名时(如果有自变量就一起附带参数，就执行文件中的语句。

所有的M文件都是普通的ASCII文件，都能通过文本编辑器创建。无论何时，如当编辑文本时，一个操作系统命令要被引用，这时从 MATLAB中使用命令！来输入和测试 M文件是很方便的，参见命令集22。

■ **例2.20**

(a) 假设某个矩阵经常被使用，它被创建并按第 2.8节存储起来，然后随时可以装入。一个替换的方法就是在一个 M文件中创建该矩阵。下面的 MATLAB命令存放在文件 **Thematrix.m**中，产生一个可以经常使用的矩阵：

```
A=[-9  -3 -16; 13  7  16; 3  3  10]
```



通过输入Thematrix，矩阵A就根据上面的那一行进行分配。通过输入下面的命令就可以显示这个矩阵的情况：

```
whos

  Name        Size            Bytes  Class

  A           3x3                72  double array

Grand total is 9 elements using 72 bytes
```

(b) 假设下列函数是存放在文件**average.m**中：

```
function y=average(A)
% 本函数计算A中所有元素的平均值，其结果为一个标量。
[m, n]=size(A);
y=sum(sum(A))/m/n;
```

如果定义了例(a)中的矩阵，输入下面的命令：

```
average_value = average(A)
```

得到的结果是：

```
average_value =
                2.6667
```

书写函数**average**的另一个方法是：

```
function y=average(A)
% 本函数计算A中所有元素的平均值，其结果为一个标量
y=mean(A(:));
```

在第6章定义了命令sum和mean，在第4.3节讲述了冒号的概念A(:)的含义。

(c) **startup.m**是一个特殊的用户定义的命令文件。如果此文件放在你的 MATLAB工作目录里，那么每当你启动MATLAB时，它就自动地运行。在这个文件里，你可以列出自己预先定义的内容和设置，如下例中，当你开始一个新奇的 MATLAB工作时， MATLAB向你致意。

```
% 我的第一个startup.m
disp('Welcome to MATLAB!')
```

有了这样一个定义的命令文件，下次启动MATLAB时就可以看到：

```
                        < M A T L A B (R) >
              (c) Copyright 1984-98 The MathWorks, Inc.
                        All Rights Reserved
                        Version 5.2.0.3084
                           Jan 17 1998

  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, type tour or visit www.mathworks.com.

 Welcome to MATLAB!
 >>
```

■

有关M文件的更多信息可参见第12.3节。



第3章 矩 阵 运 算

MATLAB中的大多数运算可以直接对矩阵应用。除了在第 2.4节中讨论的算术运算 +、－、*、^、/、\外，还有用于转置和共轭的运算符、有理数运算符和逻辑运算符。

MATLAB学生版的用户应该知道矩阵中的元素总数极限是 16384。

此外，矩阵有算术函数和逻辑函数，有些函数仅能在二维矩阵中使用。

3.1 加法和减法

如果矩阵**A**和**B**具有相同的维数, 那么就可以定义两个矩阵的和 **A**+**B**和两个矩阵的差**A**－**B**。矩阵**A**±**B**，即元素$a_{ij\cdots p}\pm b_{ij\cdots p}$。在MATLAB中，一个$m\times n$矩阵**A**和一个标量，即一个$1\times 1$矩阵$s$之间也能进行加和减运算。矩阵 **A**+$s$得到与**A**相同的维数，元素为$a_{ij}+s$。

■ **例**3.1

假设**A**和**B**定义如下：

$$\mathbf{A}=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\quad \mathbf{B}=\begin{pmatrix}5 & 6\\ 7 & 8\end{pmatrix}$$

MATLAB命令

```
Add=A+B, Sub=AB, Add100=A+100
```

得到结果：

```
Add =
     6     8
    10    12

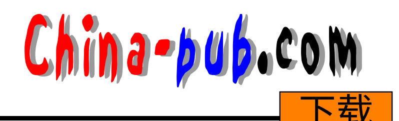
Sub =
    -4    -4
    -4    -4
Add100 =
   101   102
   103   104
```

■

3.2 乘法

如果矩阵**A**的列数等于矩阵**B**的行数，那么矩阵相乘，即**C**=**AB**，就被定义为二维矩阵。如果不是这种情况，MATLAB就返回一个错误信息。只有一个例外就是这两个矩阵之一是1×1，如一个标量，那么MATLAB是可以接受的。在MATLAB中，乘法的运算符是*，因此，命令是**C**=**A***B**。

元素c_{ij}是**A**的第i行和**B**的第j列的点积。点积的定义可参见命令集 23和附录B。矩阵**C**有与**A**相同的行数和与**B**相同的列数。



对于方阵，也定义了积**BA**，但其结果通常与**AB**不同。

■ **例**3.2

(a) 假设**A**和**B**如同例3.1，命令

```
A, B, MultAB=A*B, MultBA=B*A
```

在屏幕上显示如下的结果：

```
A =
      1      2
      3      4

B =
      5      6
      7      8

MultAB =
     19     22
     43     50

MultBA =
     23     34
     31     46
```

(b) 令x和y为：

$$\mathbf{x}=(1\quad 2\quad 3)\qquad \mathbf{y}=\begin{pmatrix}1\\10\\100\end{pmatrix}$$

命令s=x*y， M=y*x，结果为：

```
s =
   321

M =
     1      2      3
    10     20     30
   100    200    300
```

■

MATLAB也包含其他乘积。命令dot(x, y)得到具有相同元素数量的两个向量 **x**和**y**的点积，也称为标量积或内积。如果点积为零，则两个向量是正交的。如果 **A**和**B**具有相同的维数，则定义两个矩阵**A**和**B**的点积，在MATLAB中定义列方式。其结果是一个行向量，其元素是第1列、第2列等的点积，可参见附录B。

命令集23 点积

dot(x, y)	得到向量**x**和**y**的点积
dot(A, B)	得到一个长度为*n*的行向量，这里的元素是**A**和**B**对应列的点积。矩阵**A**和**B**必须是具有相同的维数*m*×*n*。多维矩阵可参见helpdesk。
dot(A, B, dim)	在*dim*数组中给出**A**和**B**的点积。



对于各具三个元素的两个向量**x**和**y**，命令cross(x, y)给出向量积或叉积，即：

$$\mathbf{x}\times\mathbf{y}=(x_2y_3-x_3y_2 \quad x_3y_1-x_1y_3 \quad x_1y_2-x_2y_1)$$

对向量**x**和**y**，向量**x**×**y**是正交的。

cross命令也可以应用于3×*n*矩阵，其结果是一个3×*n*矩阵，这里的第*i*列是**A**和**B**中的第*i*列的叉积。

命令集24 叉积

cross(x, y)	得到向量**x**和**y**的叉积。
cross(A, B)	得到一个3×*n*矩阵，其中的列是**A**和**B**对应列的叉积。矩阵**A**和**B**必须具有相同的维数3×*n*。
cross(A, B, dim)	在*dim*数组中给出向量**A**和**B**的叉积。**A**和**B**必须具有相同的维数，size(A, dim)和size(B, dim)必须是3。

■ **例3.3**

假设：

x=(1 0 0) y=(0 1 0)

命令crossprod=cross(x, ,y)得到：

```
crossprod=
        0   0   1
```

对x和y，它是正交的，即：

```
scalar1=dot(x,crossprod), scalar2=dot(y,crossprod)
```

得：

```
scalar1 =
          0

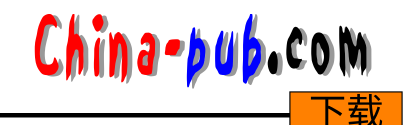
scalar2 =
          0
```

■

在MATLAB中，有一个完成二维矩阵卷积的函数。可以使用 FIR滤波器(有限脉冲响应)作为一个自变量，这部分内容在helpdesk中描述。

命令集25 矩阵的卷积

conv2(A, B)	返回矩阵**A**和**B**的二维卷积
conv2(hcol, hrow, A)	矩阵**A**与向量**hcol**列方式和向量**hrow**行方式的卷积。
conv2(…,format)	得到一个卷积的特殊形式。参数 **format**必须是下列字符串之一： ‘same’返回最接近中心的部分卷积，其维数与**A**相同。 ‘valid’仅返回不考虑边缘补零计算的部分卷积。
convn(A, B)	返回矩阵**A**和**B**的多维卷积。
convn(…, format)	得到卷积的一个特殊形式，如上所示。



Kronecker张量积可以用于创建大的矩阵，它由命令 kron(A, B)得到。如果**A**是一个$m\times n$矩阵，**B**是一个$k\times r$矩阵，那么这个命令就返回一个$m\cdot k\times r\cdot n$的矩阵。

命令集26 张量积

`kron(A,B)`	得到**A**和**B**的Kronecker张量积。

■ **例3.4**

假设：

$$\mathbf{A}=\begin{pmatrix}2&0\\-1&1\end{pmatrix}\quad \mathbf{B}=\begin{pmatrix}1&2&3\\1&0&1\end{pmatrix}$$

命令K=kron(A, B)的结果为：

```
K =
     2     4     6     0     0     0
     2     0     2     0     0     0
    -1    -2    -3     1     2     3
    -1     0    -1     1     0     1
```

■

3.3 除法

在MATLAB中，有两个矩阵除法的符号，左除 \和右除/。如果**A**是一个非奇异方阵，那么A\B和B/A对应**A**的逆与**B**的左乘和右乘，即分别等价于命令 inv(A)*B和B*inv(A)。可是，MATLAB执行它们时是不同的，如例3.5所示。A的逆，inv(A)或A^{-1}在第7.1节中介绍。

如果**A**是一个方阵，那么 X=A\B是矩阵方程**AX**=**B**的解$\mathbf{A}^{-1}\mathbf{B}$，这里的**X**具有与**B**相同的维数。在**B**=**b**是一个列向量这样一个特殊情况下，x=A\b是线性系统**AX**=**b**的解。参见第7.2节。

如果**A**是一个$m>n$的$m\times n$矩阵，X=A\B得到矩阵方程**AX**=**B**的最小二乘解，参见第7.7节。

矩阵方程**XA**=**B**的解是**X**=**B**/**A**，它等同于**(A**′**B**′**)**′,即右除可以由左除定义。这里，撇号′表示转置，这将在第3.4节中进行说明。

■ **例3.5**

(a) 设**A**和**B**如例3.1一样定义，命令

A, B, Right=B/A, Left=A\B得到：

```
A =
     1     2
     3     4

B =
     5     6
     7     8

Right =
    -1     2
    -2     3
```

```
Left =
    -3    -4
     4     5
```

如果输入`Right=B*inv(A)`和`Left=inv(A)*B`，则得到

```
Right =
   -1.0000    2.0000
   -2.0000    3.0000
Left =
   -3.0000   -4.0000
    4.0000    5.0000
```

这分别与用/和\计算的矩阵结果是一致的，但浮点格式表明它们的计算过程是不一样的。

(b) 设下列**A**和**b**：

$$\mathbf{A}=\begin{pmatrix}1&3&5\\1&2&4\\0&5&1\end{pmatrix}\qquad \mathbf{b}=\begin{pmatrix}22\\17\\13\end{pmatrix}$$

系统**Ax**=**b**的解在MATLAB中写作`x=A\b`，得到：

```
x =
     1.0000
     2.0000
     3.0000
```

(c) 使用如上的**A**和**b**,检查求解系统**Ax**=**b**的运算次数。

命令`flops(0);x=inv(A)*b; flops`给出结果：

```
ans=
       109
```

命令`flops(0); X=A\b; flops`给出结果：

```
ans=
     72
```

因此，在MATLAB中求解一个系统用左除比用逆和乘法所需的运算次数要少。命令`flops`的定义参见第2.5节。

■

3.4 转置和共轭

一个重要的运算是转置和共轭转置，它在MATLAB中用撇´表示。在课本中，这种运算经常用*和H表示

如果**A**是一个实数，那么它被转置时，第1行变成第1列，第2行变成第2列，依此类推，一个$m\times n$矩阵变为一个$n\times m$矩阵。如果矩阵是方阵，那么这个矩阵在主对角线反映出来。

如果矩阵**A**的元素a_{ij}是复数，那么所有元素也是共轭的。矩阵 **A**´在项(i, j)上含有 $\overline{a_{ji}}$ 。

如果仅希望转置，在撇号之前输入一点 .´，**A**.´表示转置，其结果与`conj(A´)` 相同。如果**A**是实数，那么**A**´与**A**.´相同。

■ **例**3.6

假设**A**和**b**与例3.5(b)相同。

`Transp=A´,Transpb=b´`,得到：

```
Transp =
      1      1      0
```




```
   3      2      5
   5      4      1

Transpb =
  22     17     13
```

■

3.5 元素操作算术运算

算术运算也可以元素与元素逐次进行。矩阵的维数要相同，可以是多维的。如果运算是由一点进行的，那么这个运算实行的是元素方式。

对于加法和减法，数组运算和矩阵运算没有差别。数组运算符是：

+ — .* ./ .\ .^

注意，.并没有列出，这个点在那种情况下具有不同的含义。这个操作符仅给出转置，与′相反，′给出了共轭转置，详见第3.4节。

■ **例**3.7

假设定义如下矩阵：

$$\mathbf{A}=\begin{pmatrix}1 & 2\\ -1 & 5\end{pmatrix}\quad \mathbf{B}=\begin{pmatrix}7 & 2\\ 1 & 0\end{pmatrix}\quad \mathbf{C}=\begin{pmatrix}1+2i & 5-2i\\ 3+i & 1+3i\end{pmatrix}$$

(a) A.∗B得：

```
ans =
    7      4
   -1      0
```

(b) B./A得：

```
ans =
    7      1
   -1      0
```

(c) B.^2得：

```
ans =
   49      4
    1      0
```

(d) A.^B得：

```
ans =
    1      4
   -1      1
```

(e) 基数是标量，而指数是一个矩阵，2.^[1 2 3 4]得：

```
ans=
   2      4      8      16
```

(f) C.′得：

```
ans =
   1.0000 + 2.0000i    3.0000 + 1.0000i
   5.0000 - 2.0000i    1.0000 + 3.0000i
```

■

可参见例13.1，它也使用了数组运算。

3.6 元素操作函数

在MATLAB中预定义的数学标准函数(见第2.4节)是基于矩阵对元素的运算。如果f是这样一个函数，**A**是带元素a_{ij}的一个矩阵，那么$f(\mathbf{A})_{ij}=f(a_{ij})$。如果元素是复数，那么根据这个函数产生的矩阵也可以是复数，矩阵的维数没有改变。

■ **例**3.8

令**A**、**B**和**C**为：

$$\mathbf{A}=\begin{pmatrix}0 & -3\\ 5 & 1\\ 4 & -6\end{pmatrix}\quad \mathbf{B}=\begin{pmatrix}\pi & 0\\ \pi/2 & \pi/4\end{pmatrix}\quad \mathbf{C}=\begin{pmatrix}1+i & -\pi i\\ 0 & 2-i\end{pmatrix}$$

(a) abs(A)得：

```
ans =
     0     3
     5     1
     4     6
```

(b) cos(B)得：

```
ans =
   -1.0000    1.0000
    0.0000    0.7071
```

(c) sin(abs(C))得：

```
ans =
    0.9878    0.0000
         0    0.7867
```

■

数组运算符和数组函数在MATLAB中十分有用，用户可以定义自己的数组函数，并把它们存放在M文件中，可参见第2.9节。

■ **例**3.9

函数sincos(x)=sin(x)cos(x)是一个非标准MATLAB函数，可是，你可以定义自己的函数**sincos**并存放在文件**sincos.m**中。

```
function y=sincos(x)
y=sin(x).*cos(x);
```

可如下调用*sincos*:

```
y1 = sincos(pi), y2 = sincos([0 pi/4 pi/2])

y1 =
  -1.2246e-16

y2 =
         0    0.5000    0.0000
```

看到，应该为0值的y1是一个十分小的数。事实上， *eps*是较大的。如果用一个向量作为一个自变量来调用 *sincos*，因为`sin`和`cos`返回向量，所以其结果是一个向量。当绘制函数图形时，这是十分有用的。

■

M文件的应用可参见第12章和第13章。

3.7 矩阵的乘方与函数

对于二维方阵，**A**的p次乘方可以用A^p实现。如果p是一个正整数，那么这个幂可以由许多矩阵乘法运算定义。对于 p=0，得到与**A**维数相同的同一个矩阵；当 p<0时，如果$\mathbf{A}^{-1}$存在，可定义A^p，它是与inv(A)^(-p)相同。

象exp(A)和sqrt(A)那样的MATLAB表达式可视为数组运算(参见第3.6节)，即它们是对**A**中元素逐个运算。

MATLAB也能处理方阵函数。例如$\mathbf{A}^{1/2}$(**A**的平方根)或$e^{\mathbf{A}}$。举例如下：

$$e^{\mathbf{A}} = I + \mathbf{A} + \frac{\mathbf{A}^2}{2!} + \frac{\mathbf{A}^3}{3!} + \cdots$$

命令集27 矩阵函数

命令	说明
`expm(A)`	使用Pade近似法计算e^{A}，这是一个内部函数。
`expm1(A)`	使用一个M文件和与内部函数相同的算法计算 e^{A}。
`expm2(A)`	使用泰勒级数计算e^{A}。
`expm3(A)`	使用特征值和特征向量计算 e^{A}。
`logm(A)`	计算**A**的对数。
`sqrtm(A)`	计算$\mathbf{A}^{1/2}$。当**A**是对称正定阵时，平方根是唯一的。
`funm(A, fcn)`	计算由字符串 **fcn**指定的**A**的矩阵函数，参见第 5.1.4节。字符串**fcn**可以是任意的基本函数，如 sin、cos等等，参见第2.4节。例如， expm(A)=funm(A,‘exp’) 。
`[F, E]=funm(A, fcn)`	计算如上矩阵函数，但返回结果矩阵**F**和剩余近似值矩阵**E**。
`polyvalm(p,A)`	估算矩阵**A**的一个多项式。向量**p**含有多项式的系数。详见第10.1节。

很重要的一点是要区别`expm`和`exp`、`logm`和`log`等等。

■ **例3.10**

假设：

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$$

比较`exp`和`expm`：

```
Elementwise=exp(A), Operatorwise=expm(A)
```

得：

```
Elementwise =
     2.7183     1.0000
     1.0000     7.3891
```

```
Operatorwise =
    2.7183         0
         0    7.3891
```

■

3.8　关系运算符

MATLAB有用于比较矩阵的六个关系运算符，也可以对矩阵与一个标量进行比较，即矩阵中的每个元素与标量进行比较。

关系运算符如下：

<	小于
<=	小于等于
>	大于
>=	大于等于
==	等于
~=	不等于

关系运算符比较对应的元素，产生一个仅包含 1和0的具有相同维数的矩阵。其元素是：

1　比较结果是真

0　比较结果是假

在一个表达式中，算术运算符优先级最高，其次是关系运算符，最低级别是逻辑运算符。圆括号可以改变其顺序。

■ **例**3.11

(a) 对预定义变量pi的值和通过命令rat获得的pi的近似值进行比较。

```
[t, n]=rat(pi), piapprox=t/n;
format long, piapprox, pi, piapprox==pi
```

得：

```
t =
   355

n =
   113

piapprox =
   3.14159292035398

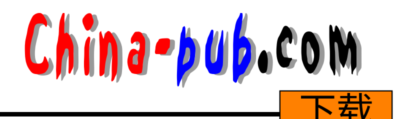
ans =
   3.14159265358979

ans =
     0
```

(b) 假设：

$$\mathbf{A}=\begin{pmatrix}1&2&4\\1&1&1\\2&3&1\end{pmatrix}\quad \mathbf{B}=\begin{pmatrix}2&2&2\\2&2&2\\2&2&2\end{pmatrix}$$

A中的元素有大于**B**中对应的元素吗？




```
Greater=A>B
```

得：

```
Greater =
      0      0      1
      0      0      0
      0      1      0
```

即**A**中的项(1，3)和(3，2)的值大于**B**中对应项的值。

(c) 令**A**如例(b)中假设，**A**中的元素有大于1的吗？

```
GreaterThanOne=A>1
```

得：

```
GreaterThanOne =
      0      1      1
      0      0      0
      1      1      0
```

■

3.9 逻辑运算符

在MATLAB中有四种逻辑运算符：

& 与

| 或

~ 非

xor 异或

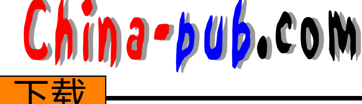
逻辑运算符的运算优先级最低。在一个表达式中，关系运算符和算术运算符的运算级别要高于逻辑运算符。

xor和**or**之间的差别在于：表达式中至少有一个是真，那么 **or**是真；**xor**是表达式中有一个是真但不能两者均为真时才为真。

运算符&和|比较两个相同维数的矩阵，如同前一节一样，它也能使一个标量与一个矩阵进行比较。逻辑运算符是按元素比较的。零元素表示逻辑值假，任何其他值的元素表示逻辑值真。其结果是一个包含1和0的矩阵。

命令集28 逻辑运算符

命令	说明
A&B	返回一个与**A**和**B**相同维数的矩阵。在这个矩阵中，**A**和**B**对应元素都为非零时，则对应项为1；有一个为零的项则为0。
A|B	返回一个与**A**和**B**相同维数的矩阵。在这个矩阵中，**A**和**B**对应元素只要有一个为非零，则对应项为1；两个矩阵均为零时，则为0。
~A	返回一个与**A**和**B**相同维数的矩阵。在这个矩阵中，**A**是零时，则对应项为1；**A**是非零时，则对应项为0。
xor(A, B)	返回一个与**A**和**B**相同维数的矩阵。在这个矩阵中，如果 **A**和**B**均为零或均为非零时，则对应项为0；如果**A**或**B**是非零但不是两者同时为非零时，则对应项为1。



3.10 逻辑函数

在MATLAB中有几个逻辑函数。在以下定义的函数中，假设 **A**是一个$m \times n$的矩阵， **x**是一个向量。

在一些计算中，很重要的一点是要在给定的矩阵中以一定的特征定位。例如，在部分选主元的高斯消去法中，必须在工作列中寻找最大的项。 MATLAB命令`find`可以用于这种情况。

命令集29 查找非零元素

`find(x)`	返回一个**x**中包含非零元素的下标的向量。如果所有的元素都是零，那么返回一个空矩阵，即[]。
`find(A)`	返回一个长的列向量，表示 **A**中包含非零元素的下标向量。下述命令更可取。
`[u, v]=find(A)`	返回向量**u**和**v**，它们包含 **A**中的非零元素的下标，即 **A**中元素(u_k, v_k)为非零。
`[u,v,b]=find(A)`	返回包含**A**中非零元素的下标向量 **u**和**v**以及一个包含对应非零元素的向量。**A**中元素(u_k, v_k)为非零并且能在b_k中找到。

■ **例**3.12

假设**x**和**A**是：

$$\mathbf{x} = (3 \quad -4 \quad 0 \quad 6.1 \quad 0) \quad \mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 4 \end{pmatrix}$$

(a) `ind=find(x), indcol=find(A)` 得

```
ind =
     1     2     4

indcol =
     1
     4
```

即向量**x**中元素1、2和4是非零值。要获得`indcol`，也可以输入：

```
find(y)
```

(b)命令find可以与关系运算符一起使用，这样使命令更有用。例如,index=find(x>0.5)返回：

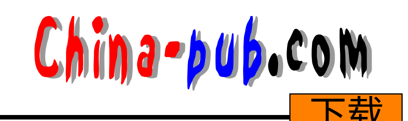
```
index=
     1     4
```

如果输入`greaterThan=x(index)`, 得到：

```
greaterThan=
  3.0000    6.1000
```

这就是用向量**index**寻找**x**中所有大于0.5的元素。

如果仅想知道**x**中有多少大于0.5的元素时，可以输入：`length(find(x>0.5))`。针对上题数据，得到：




```
ans=
   2
```

(c) 要获得**A**中所有非零元素的索引，输入：

```
[index1, index2]=find(A)
```

得：

```
index1 =
     1
     2

index2 =
     1
     2
```

即元素(1，1)和(2，2)是非零值。

■

MATLAB有any和all两个函数，用于测试矩阵和向量的逻辑条件。其结果是逻辑值，0或1，真或假。它们在if语句中有特殊的用途，详见第12.1节。

命令集30 逻辑函数(一)

any(x)	如果**x**中的有一个元素为非零值，那么返回1；否则，返回0。
any(A)	对**A**进行列运算，根据相应列是否包含非零元素，返回一个带1和0的行向量。
all(x)	如果所有的元素都是非零值，返回1；否则，返回0。
all(A)	对**A**进行列操作，根据相应列是否所有元素都为非零值，返回带1和0的一个行向量。

如果这两个函数之一对一个矩阵进行两次操作，例如 any(any(A))和all(all(A))则返回一个标量1或0。

■ 例3.13

(a) 如果all(x<=5)返回1，则实向量**x**中所有的元素都小于或等于5。如果返回0，则至少有一个元素大于5。如果all(all(A<=5))返回1，则一个矩阵**A**的所有元素小于或等于5。

(b) 对一个实方阵**A**，如果all(all(A==A'))返回1，则**A**是对称的。

(c) 如果any(any(tril(A,−1)))返回0，则方阵**A**是上三角阵。否则，在**A**中的下三角阵中至少有一个非零元素。一个等价的命令是 all(all(A==triu(A)))，如果**A**是上三角阵，它就返回1。

■

也有下列逻辑函数：

命令集31 逻辑函数(二)

isnan(A)	返回一个维数与**A**相同的矩阵，在这个矩阵中，对应**A**中有'NaN'处为1，其他地方为0。

isinf(A)	返回一个维数与 **A**相同的矩阵，在这个矩阵中，对应 **A**中有‘inf’处为1，其他地方为0。
isempty(A)	如果**A**是一个空矩阵，返回1；否则返回0。
isequal(A, B)	如果**A**和**B**是相同的，即有相同的维数和相同的内容，则返回1。
isreal(A)	如果**A**是一个不带虚部的实矩阵，则返回1；否则，返回零。
isfinite(A)	返回一个与**A**维数相同的矩阵。在这个矩阵中，**A**中元素是有限的，则对应元素为1；否则，为零。

也有针对字符串和其他数据类型的逻辑函数，详见第 5章。



第4章　创建新矩阵

在2.2节中讨论了矩阵的定义和分配。但是也还有可能要建立新的矩阵，例如通过函数返回一个新的矩阵或者使用已存在的矩阵。

4.1　建立新矩阵

建立1矩阵使用ones命令，这种矩阵的元素全部都是1。相应的建立0矩阵使用zeros命令，这种矩阵的元素全部都是 0。单位矩阵的对角线元素全部是 1，而其他元素全部是 0。建立单位矩阵使用eye命令。在矩阵乘方运算中，*n*阶的单位矩阵就相对应于在标量运算中的数字 1。

命令集32　**1矩阵、零矩阵和单位矩阵**

ones(n)	建立一个$n \times n$的1矩阵。
ones(m,n,...,p)	建立一个$m \times n \times \cdots \times p$的1矩阵。
ones(size(A))	建立一个和矩阵**A**同样大小的1矩阵。
zeros(n)	建立一个$n \times n$的0矩阵。
zeros(m,n,...,p)	建立一个$m \times n \cdots \times p$的0矩阵。
zeros(size(A))	建立一个和矩阵**A**同样大小的0矩阵。
eye(n)	建立一个$n \times n$的单位矩阵。注意eye命令只能用来建立二维矩阵。
eye(m, n)	建立一个$m \times n$的单位矩阵。注意eye命令只能用来建立二维矩阵。
eye(size(A))	建立一个和矩阵**A**同样大小的单位矩阵。

■ **例4.1**

命令组：

```
OneMatrix  = ones(3,4,2)
ZeroMatrix = zeros(size(OneMatrix))
Identity   = eye(2)
Identity23 = eye(2,3)
Identity32 = eye(3,2)
```

运行上述命令后在屏幕上会得到以下结果：

```
OneMatrix(:,:,1) =
     1     1     1     1
     1     1     1     1
     1     1     1     1

OneMatrix(:,:,2) =
     1     1     1     1
     1     1     1     1
     1     1     1     1

ZeroMatrix(:,:,1) =
```

```
      0     0     0     0
      0     0     0     0
      0     0     0     0

ZeroMatrix(:,:,2) =
      0     0     0     0
      0     0     0     0
      0     0     0     0

Identity =
      1     0
      0     1

Identity23 =
      1     0     0
      0     1     0

Identity32 =
      1     0
      0     1
      0     0
```

■

矩阵中的所有元素都是随机数，这样的矩阵称为随机矩阵。可以用 rand命令来产生在0～1之间均匀分布的随机数。也可以用MATLAB中的randn命令来产生服从零均值、单位方差正态分布的随机数。

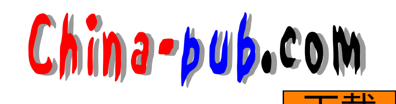
命令集33 随机数和随机矩阵

`rand`	产生在0～1之间均匀分布的随机数；每调用一次给一个新的数值。
`rand + i*rand`	产生一个复数随机数。
`rand(n)`	产生一个$n \times n$的矩阵，其元素为0～1之间均匀分布的随机数。
`rand(m,n,...,p)`	产生一个$m \times n \times \cdots \times p$的矩阵，其元素为0～1之间均匀分布的随机数。
rand	产生零均值、单位方差的正态分布随机数。
`randn(n)`	产生一个$n \times n$的矩阵，其元素为零均值、单位方差的正态分布随机数。
`randn(m,n,...,p)`	产生一个$m \times n \times \cdots \times p$的矩阵，其元素为零均值、单位方差的正态分布随机数。

MATLAB 5使用一种新的随机数发生器，可以设置几个随机数的种子。它能够产生在闭区间$[2^{-53}, 1-2^{-53}]$上所有的浮点数。理论上它能够产生 $2^{1492} \cong 10^{449}$多个不重复的数。而**MATLAB 4**中的随机数发生器只能使用一个随机数的种子。

命令集34 随机数种子

`rand('state')`	返回一个有35个元素的向量，其中包含随机状态发生器的当前状态。
`rand('state',s)`	设置随机种子发生器的状态为s。



`rand('state',0)`	设置随机种子发生器为它的原始状态。
`rand('state', j)`	设置随机种子发生器为它的第 *j*种状态，*j*为整数。
`rand('state', sum(100*clock))`	使用clock命令(见命令集15)，使得随机种子发生器在每个不同时刻都设置为一种不同的状态。
`rand('seed',arg)`	使用**MATLAB 4**中的随机种子发生器，见帮助可以得到更多的信息。
`randn('state')`	返回一个有两个元素的向量，其中包含正态随机种子发生器的状态。
`randn('state',arg)`	根据**arg**设置正态随机种子发生器，见rand。

■ **例**4.2

(a) 例如，随机种子发生器可以给出以下的结果。这里只列出了这个状态向量（35个元素）中的前五个元素的值。

```
astate = rand('state'); astate(1:5), Random = rand(2,3)
ans =
     0.6923
     0.1646
     0.5676
     0.3609
     0.8557
Random =
     0.4565     0.8214     0.6154
     0.0185     0.4447     0.7919
```

(b)为了避免总是从相同的随机种子开始而得到相同的随机数序列，可以使用 **MATLAB**中的clock函数。

```
rand('state',sum(100*clock)); R = rand('state'); R(1:5)
ans =
     0.8010
     0.4701
     0.5052
     0.0707
     0.4643
```

clock命令定义在2.5节中。

■

在MATLAB中还有利用已存在的矩阵建立新矩阵的命令。假设矩阵 **A**是$m \times n$的矩阵，**x**是一个有n个元素的向量。用命令集35中的命令diag来生成一个新的矩阵。

命令集35　从已存在的矩阵中生成新的矩阵(一)

`diag(A)`	生成一个由矩阵 **A**主对角线元素组成的列向量。主对角线总是从矩阵左上角开始。对于方阵来说它结束于矩阵的右下角。

`diag(x)`	生成一个n维的方阵，它的主对角线元素值取自向量 **x**，其余元素的值都为0。
`diag(A,k)`	生成一个由矩阵 **A**第k条对角线的元素组成的列向量。 k=0为主对角线；k<0为下第k对角线；k>0为上第k对角线。
`diag(x,k)`	生成一个$(n+\text{abs}(k))\times(n+\text{abs}(k))$维的矩阵，该矩阵的第 k条对角线元素取自向量**x**，其余元素都为零。关于参数k可参考上个命令。

■ **例**4.3

假设：

$$\mathbf{A}=\begin{pmatrix}1&2&3&4\\5&6&7&8\\9&10&11&12\\13&14&15&16\end{pmatrix}\qquad \mathbf{x}=(-5\quad -10\quad -15)$$

(a) 命令`diag_element=diag(A)`给出：

```
diag_element =
      1
      6
     11
     16
```

(b) `Diag_matrix=diag(diag(A))`返回：

```
Diag_matrix =
      1      0      0      0
      0      6      0      0
      0      0     11      0
      0      0      0     16
```

(c) 命令`Dmatrixx=diag(x)`或者`Dmatrixx=diag(x')` 给出：

```
Dmatrixx =
     -5      0      0
      0    -10      0
      0      0    -15
```

(d) 如果输入`superDiagElement=diag(A,2)`，那么输出：

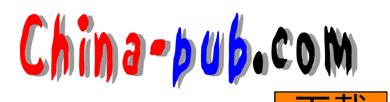
```
superDiagElement =
     3
     8
```

(e) `NewMatrix=diag(diag(A,2))`返回：

```
NewMatrix =
     3   0
     0   8
```

注意，该矩阵的大小由命令`diag(A, 2)`生成的向量决定。

(f) `SuperDiagonalMatrix =diag(diag(A,2),2)`返回下列矩阵：




```
SuperDiagonalMatrix =
      0       0       3       0
      0       0       0       8
      0       0       0       0
      0       0       0       0
```

矩阵A的上第2对角线的长度为2，因此建立的矩阵大小为4×4。 ■

在MATLAB中使用命令triu和tril来建立三角矩阵。

命令集36 从已存在的矩阵中生成新的矩阵(二)

`triu(A)`	生成一个和A大小相同的上三角矩阵。该矩阵的主对角线及以上元素取自A中相应元素，其余元素都为零。
`triu(A,k)`	生成一个和A大小相同的上三角矩阵。该矩阵的第k条对角线及以上元素取自A中相应元素，其余元素都为零。命令triu(A,0)等同于命令triu(A)。
`tril(A)`	生成一个和A大小相同的下三角矩阵。该矩阵的主对角线及以下元素取自A中相应元素，其余元素都为零。
`tril(A,k)`	生成一个和A大小相同的下三角矩阵。该矩阵的第k条对角线及以下元素取自A中相应元素，负数k表示主对角线下的对角线。其余元素都为零。命令tril(A,0)等同于命令tril(A)。

对于每一个方阵A都有下列关系：

```
A=triu(A) + tril(A) - diag(diag(A))
```

严格的上三角矩阵A应该使用triu(A,1)来定义；而严格的下三角矩阵A则用tril(A,−1)来定义。因此，对于每一个方阵A都有下列关系：

```
A=triu(A, 1) + tril(A,-1) + diag(diag(A))
```

当通过迭代的方法来求解线性方程系统 (例如Gauss-Seidel, Jacobi 或者Successive Over Relaxation(SOR))时，以这种方式来分解矩阵是很重要的。

■ 例4.4

假设：

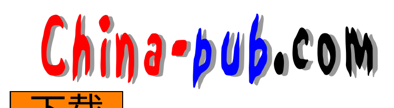
$$\mathbf{B} = \begin{pmatrix} 9 & 8 & 7 & 6 \\ 1 & 3 & 0 & 7 \\ -4 & 7 & 1 & 9 \end{pmatrix}$$

(a) `UpperTriangular =triu(B)`返回：

```
UpperTriangular =
      9       8       7       6
      0       3       0       7
      0       0       1       9
```

(b) `LowerTriangular =tril(B,-1)`返回：

```
LowerTriangular =
      0       0       0       0
      1       0       0       0
     -4       7       0       0
```



还有一些命令可以用来变换矩阵结构。

■

命令集37 矩阵旋转和矩阵变维

命令	说明
fliplr(A)	通过二维矩阵**A**的行元素按照$b_{ij}=a_{i,n-j+1}$交换位置生成一个新矩阵。这里的‘lr’是‘left-right’的缩写。
flipud(A)	通过二维矩阵**A**的列元素按照$b_{ij}=a_{m-i+1,j}$交换位置生成一个新矩阵。这里的‘ud’是‘up-down’的缩写。
flipdim(A, dim)	生成一个在dim维矩阵**A**内的元素交换位置的多维矩阵。命令flipdim(A, 1)等同于命令flipud(A)，命令flipdim(A, 2)等同于命令fliplr(A)。
rot90(A)	生成一个由矩阵**A**逆时针旋转90°而得的新阵。也就是将矩阵**A**中的左上角的元素和右下角的元素交换位置，也可见13.5节。
rot90(A, k)	生成一个由矩阵**A**逆时针旋转k×90°而得到的新阵，也可见13.5节。
reshape(A,m,n,...p)	生成一个m×n×…×p维的矩阵，它的元素以线性索引的顺序(见图2-2)从矩阵**A**中取来。如果矩阵**A**中没有m×n×…×p个元素，将返回一个错误信息。
repmat(A,[m n ...p])	创建一个和矩阵A有相同元素的m×n×…×p块的多维矩阵。
repmat(x,[m n ...p])	创建一个m×n×…×p的多维矩阵，所有元素的值都为标量x。使用该命令要比用命令x*ones([m n ...])来创建同一个大矩阵的速度要快。
shiftdim(A,n)	矩阵的列移动n步。n为正数，矩阵向左移；n为负数，向右移。
squeeze(A)	返回没有空维的矩阵**A**。
cat(dim,A,B)	将矩阵**A**和**B**组合成一个dim维的多维矩阵。
permute(A,order)	根据向量**order**来改变矩阵**A**中的维数顺序。
ipermute(A,order)	进行命令permute的逆变换。命令ipermute(permute(A, order), order) 得到的结果就是矩阵**A**本身。

■ **例4.5**

(a) 假设有如例4.1中的多维矩阵**OneMatrix**，使用命令B=reshape(OneMatrix, 3, 8)可以使它变维而成为二维矩阵，结果如下：

```
B =
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
```

(b) 为了在矩阵**B**中增加一层零元素，可以先使用命令C=zeros(3,8)创建一个零阵。然后通过下列命令来得到一个合并的矩阵：

```
D = cat (3, B, C)

D(:,:,1) =
       1     1     1     1     1     1     1     1
       1     1     1     1     1     1     1     1
       1     1     1     1     1     1     1     1

D(:,:,2) =
       0     0     0     0     0     0     0     0
       0     0     0     0     0     0     0     0
       0     0     0     0     0     0     0     0
```

(c) 为了快速对矩阵**D**进行变维以便它可以响应从命令`cat(3,C,B)`返回的结果，可以使用：

```
flipdim(D,3)

ans(:,:,1) =
       0     0     0     0     0     0     0     0
       0     0     0     0     0     0     0     0
       0     0     0     0     0     0     0     0

ans(:,:,2) =
       1     1     1     1     1     1     1     1
       1     1     1     1     1     1     1     1
       1     1     1     1     1     1     1     1
```

(d) 使用命令`permute`和`shiftdim`对矩阵变维的结果如下：

```
size(shiftdim(D,2))

ans =
       2     3     8

size(permute(D,[2 1 3]))

ans =
       8     3     2
```

在**MATLAB**中可以通过增加元素、行和列将一个矩阵或者向量进行扩展。由于 **MATLAB** 可以自动地改变矩阵的大小，所以使用已存在的矩阵的一部分来创建一个新矩阵是很容易的，这在许多应用中都很有用。

从已存在的矩阵中建立一个矩阵就和定义一个新矩阵一样。元素用空格或逗号分隔，行用分号或回车分隔；见2.2节。在4.3节中给出了其相反过程，从大矩阵中定义子矩阵。

■

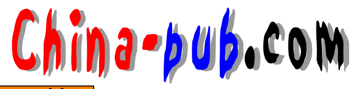
■ **例**4.6

假设下列矩阵已经定义为：

$$\mathbf{A}=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\quad \mathbf{B}=\begin{pmatrix}5 & 6\\ 7 & 8\end{pmatrix}\quad \mathbf{x}=(9\quad 10)\quad \mathbf{y}=\begin{pmatrix}11\\ 12\end{pmatrix}\quad \mathbf{z}=(13\quad 14)$$

(a) 有几种方式可以将向量**x**扩展成1×4。假设想要的新向量是：

xnew=(9　10　0　5)



下列的三种方法都可以给出想要的结果：

```
(i) xnew = x; xnew(3) = 0; xnew(4) = 15;
(ii) xnew = [x 0 15];
(iii) temp = [0 15]; xnew = [x temp];
```

(b) 以下两种方法可以对矩阵**A**扩展一个新行，如向量**z**：

```
(i) Anew1 = [A; z];
(ii) Anew1 = [A; [13 14]];
```

它们在屏幕上显示的结果如下：

```
Anew1 =
     1     2
     3     4
    13    14
```

有时还可以对矩阵添加多个新行：

```
Anew2 = [A; x; z; [0 0]]
Anew2 =
     1     2
     3     4
     9    10
    13    14
     0     0
```

对矩阵**A**扩展一个新列，如**y**，可以这样做：

`Anew3=[A  y]` 或者 `Anew3=[A [11; 12]]`

它们在屏幕上显示的结果如下：

```
Anew3 =
     1     2    11
     3     4    12
```

扩展一个矩阵的操作是相似的。输入命令：

`ABvert=[A; B]` 和 `ABhoriz=[A B]`

就可以得到：

```
ABvert =
     1     2
     3     4
     5     6
     7     8

ABhoriz =
     1     2     5     6
     3     4     7     8
```

对于**ABvert**来说，它的列数一定等于矩阵 **A**和**B**的列数；而对于 **ABhoriz**来说，它的行数一定等于矩阵**A**和**B**的行数。

■

为了生成规则的矩阵块可以下列的方式使用命令 `repmat`。

■ **例**4.7

(a) 命令`repmat([1 0; 0 1],3,3)`将返回：

```
ans =
     1     0     1     0     1     0
     0     1     0     1     0     1
     1     0     1     0     1     0
     0     1     0     1     0     1
     1     0     1     0     1     0
     0     1     0     1     0     1
```

(b) 在例3.2中使用命令`repmat([1 0],1,5)`得到：

```
ans =
     1     0     1     0     1     0     1     0     1     0
```

(c) 如果要创建一个所有元素都是同一个值的矩阵，可以使用命令`repmat(42,[22])`来创建，给出的结果如下：

```
ans =
    42    42
    42    42
```

■

4.2 空矩阵

在**MATLAB**中对空矩阵的定义是`A=[]`。有时创建一个多维的矩阵，但是这个矩阵中可能有几维是空的，比如0×1×0矩阵。也可参见命令集31中的命令`isempty`。

■ **例**4.8

要创建一个大小为1-2-0-0-2的矩阵，可以使用命令`zeros(1,2,0,0,2)`，结果为：

```
ans =
   Empty array: 1-by-2-by-0-by-0-by-2
```

空的行向量和列向量用命令`zeros`来定义：

```
rowvect = zeros(1,0)

rowvect =
   Empty matrix: 1-by-0

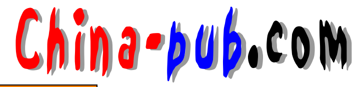
colvect = zeros(0,1)

colvect =
   Empty matrix: 0-by-1
```

现在这些向量的大小为：

```
size(rowvect), size(colvect)

ans =
     1     0
ans =
     0     1
```



一个空矩阵可以这样来创建：

```
A = []

A =
     []
```

通过命令whos来查看在内存中的驻留变量的详细信息：

```
  Name          Size        Bytes        Class

  A             0x0             0      double array
  colvect       0x1             0      double array
  rowvect       1x0             0      double array

Grand total is 0 elements using 0 bytes
```

■

一些函数对空矩阵操作返回一个常量，在编写程序时这常常是有用的。在命令集 38中**E**是一个空矩阵，为了清除矩阵中的空维可以使用命令 squeeze。

命令集38　空矩阵函数

squeeze(A)	返回没有空维的矩阵**A**。
sum(E)	返回0。
prod(E)	返回1。
max(E)	返回**E**。
min(E)	返回**E**。

4.3　向量和子矩阵的生成

在MATLAB中可以使用冒号‘ : ’来代表一系列数值。有时也使用它来定义一个子矩阵。我们先给出用冒号来定义向量的方法。

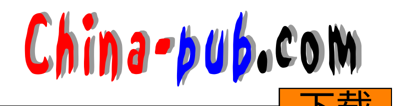
命令集39　数字序列(一)

i:k	创建从i开始、步长为1、到k结束的数字序列，即i ,i+1，i+2，…，k。如果i>k，MATLAB则返回一个空矩阵，也就是 []。数字i和k不必是整数，该序列的最后一个数是小于或等于k。
i:j:k	创建从i开始、步长为1、到k结束的数字序列，即i，i+j，i+2j，…，k。对于j=0，则返回一个空矩阵。数字i、j和k不必是整数，该序列的最后一个数是小于或等于k。

■例 4.9

(a) 如果输入vect=2:7或者vect=2:7.7，MATLAB返回相同结果：

```
vect =
     2     3     4     5     6     7
```



(b) 负步长：`vect2=6:－1:1`，结果为：

```
vect2 =
     6     5     4     3     2     1
```

(c) 实数：`realVect=1.2:－0.8:－3.2`，结果为：

```
realVect =
    1.2000    0.4000   -0.4000   -1.2000   -2.0000   -2.8000
```

注意最后一个数值为－2.8。

(d) 命令`realVect2=0:pi/4:pi`，结果为：

```
realVect2 =
         0    0.7854    1.5708    2.3562    3.1416
```

(e) 冒号可以用来定义矩阵：

```
Mat1=[2:4 0.1:1:2.1; 1:6]
```

结果为：

```
Mat1 =
    2.0000    3.0000    4.0000    0.1000    1.1000    2.1000
    1.0000    2.0000    3.0000    4.0000    5.0000    6.0000
```

(f) 冒号能够生成函数表，比如sine：

```
a = 0.0; b = 2*pi; n = 11; x = (a:(b-a)/(n-1):b)';
y = sin(x); Ftable = [x y]
```

结果为：

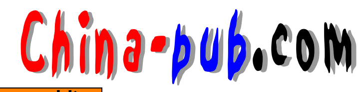
```
Ftable =
         0         0
    0.6283    0.5878
    1.2566    0.9511
    1.8850    0.9511
    2.5133    0.5878
    3.1416    0.0000
    3.7699   -0.5878
    4.3982   -0.9511
    5.0265   -0.9511
    5.6549   -0.5878
    6.2832   -0.0000
```

■

还有一些预定义函数也可以用来创建线性序列和逻辑序列。在绘图函数中这些序列都是有用的。

命令集40 数字序列(二)

`linspace(a,b)`	在区间[a, b]上创建一个有100个元素的向量，这100个数把整个区间线性分隔。
`linspace(a,b,n)`	在区间[a, b]上创建一个有n个元素的向量。这个命令和冒号表示形式相近，但是它直接定义了数据的个数。



`logspace(a,b)`	在区间$[10^a,10^b]$上创建一个有50个元素的向量，这50个数把整个区间对数分隔。特例：如果`b=pi`，则函数返回一个在区间$[10^a,\pi]$上服从对数分布的向量。
`logspace(a,b,n)`	在区间$[10^a,10^b]$上创建一个有n个元素的向量，这n个数把整个区间对数分隔，特例情况同上。

如果从矩阵**C**中抽取行和/或列组成矩阵**D**，那么**D**就称为**C**的子阵，**C**中的行和列也可以称为**C**的子阵。所以一个矩阵可以有许多子阵。这可以推广到多维数组中去。在命令集41中列出了对二维数组操作的命令。

要取出其中一维的最后一个元素值，可以用值`end`来取。例如，A是一个4×3×2的数组，`A(end,2,1)`就可以得到元素a_{421}的值，`A(end,end,end)`得到元素a_{432}的值。

当用冒号来定义矩阵A的子阵时，要使用在命令集41中列出的表达式。

命令集41 定义子阵

`A(i,j,···,k)`	返回多维数组**A**中下标为$(i, j, \cdots, k)$的元素值，也可参见2.3节。
`A(:,j)`	返回二维矩阵**A**中第j列列向量。
`A(i,:)`	返回二维矩阵**A**中第i行行向量。
`A(:,j:k)`	返回由二维矩阵**A**中的第j列，第j+1列，直到第k列列向量组成的子阵。
`A(i:k,:)`	返回由二维矩阵**A**中的第i行，第i+1行，直到第k行行向量组成的子阵。
`A(i:k,j:l)`	返回由二维矩阵**A**中的第i行到第k行行向量和第j列到第l列列向量组成的子阵。
`A(:,:,···,:)`	返回矩阵**A**本身。
`A(:)`	将矩阵**A**中的每列合并成一个长的列向量。
`A(j:k)`	返回一个行向量，其中的元素为A(:)中的从第j个元素到第k个元素。
`A([j1 j2···])`	返回一个行向量，其中的元素为A(:)中的第j1、j2···元素。
`A(:,[j1 j2··])`	返回矩阵**A**的第j1列、第j2列等的列向量。
`A([i1 i2···]:,)`	返回矩阵**A**的第i1行、第i2行等的行向量。
`A([i1 i2··], [j1 j2···])`	返回矩阵第i1行、第i2行等和第j1列、第j2列等的元素。

也可参见`help colon`。

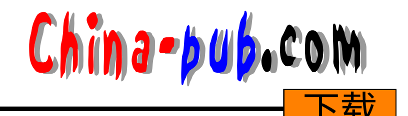
■ **例**4.10

假设矩阵**Ftable**如例4.8(f)中一样定义。

(a) 语句`Submatrix=Ftable(2:4,:)`输出的结果为：

```
Submatrix =
    0.6283    0.5878
    1.2566    0.9511
    1.8850    0.9511
```

也就是它的每一列是从矩阵**Ftable**的第2到第4行。



(b) 为了使得从一个$i \times j \times k$的立方体原点到3×3×3的立方体的中心最远，可以使用的命令是：

```
A(end-2: end , end-2: end , end-2: end )
```

(c) 冒号表达式能够与关系运算符一起使用，见3.8节。通过下面简短的命令可以挑选出矩阵**Ftable**第2列中大于0的元素所在的行向量：

```
Selected=Ftable(Ftable(:, 2) >0 ,: )
```

所得的结果为：

```
Selected =
     0.6283     0.5878
     1.2566     0.9511
     1.8850     0.9511
     2.5133     0.5878
     3.1416     0.0000
```

■

利用冒号表达式可以写出复合表达式，用help list可得更多的信息。

4.4 MATLAB中的特殊矩阵

在4.1节提到的零矩阵、单位矩阵和1矩阵都属于特殊矩阵，在同一节中还列出了对随机矩阵的操作命令。

另外，MATLAB中还有一些命令用于生成试验矩阵。希尔伯特(Hilbert)矩阵，也称H阵，其元素为$h_{ij}=1/(i+j-1)$。由于它是一个条件数差的矩阵，所以将它用来作为试验矩阵；见7.6节。

命令集42 希尔伯特矩阵

hilb(n)	生成一个$n \times n$的希尔伯特矩阵。
invhilb(n)	生成一个$n \times n$的希尔伯特矩阵的逆矩阵，其元素都为整数。

■ **例4.11**

如果输入H=hilb(3)，Hinv=invhilb(3)，MATLAB就会相应地输出：

```
H =
     1.0000     0.5000     0.3333
     0.5000     0.3333     0.2500
     0.3333     0.2500     0.2000

Hinv =
       9    -36     30
     -36    192   -180
      30   -180    180
```

可以看出希尔伯特矩阵和它的逆矩阵都是对称矩阵；见附录B。

■

托普利兹(Toeplitz)矩阵由两个向量来定义，一个行向量和一个列向量。对称的托普利兹矩阵由单一向量来定义。

命令集43 托普利兹矩阵

toeplitz(k,r)	生成一个非对称的托普利兹矩阵，将k作为第1列，将r作为第1行。



	其余的元素与左上角相邻元素相等。
toeplitz(c)	用向量**c**生成一个对称的托普利兹矩阵。

■ **例**4.12

已知 x=[1 2 3 4] y=[9 8 7 6] 那么

```
Toepmatrix1 = toeplitz(x,y), Toepmatrix2 = toeplitz(y,x)
```

给出结果为：

```
Column wins diagonal conflict.
Toepmatrix1 =
      1     8     7     6
      2     1     8     7
      3     2     1     8
      4     3     2     1

Column wins diagonal conflict.
Toepmatrix2 =
      9     2     3     4
      8     9     2     3
      7     8     9     2
      6     7     8     9
```

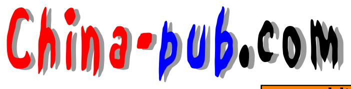
■

可以通过命令gallery来调用特殊矩阵库。输入help gallery可以找到哪些矩阵族是可用的，输入help private/family可以得到特殊矩阵族的有关信息。旧版本中保留的有关特殊命令集应该避免使用。

在下面的命令集44中给出了MATLAB中其他特殊矩阵。

命令集44 其他特殊矩阵

compan(p)	生成一个**p**多项式的友矩阵，也就是它的特征多项式是 **p**。**p**是一个包含多项式系数的向量，见10.1节。
gallery(n)	生成一个在数字分析中有名的 $n \times n$试验矩阵。比如，只有 n=3和n=5时该矩阵才存在：n=3时是一个条件数差的矩阵；n=5时是一个有意义的特征值问题矩阵。
gallery family	从family族中返回一个矩阵；见表4-1。
hadamard(k)	返回一个阶数为 $n=2^k$的Hadamard矩阵。只有当 n能被4整除时Hadamard矩阵才存在。
hankel(x)	返回一个由向量**x**定义的Hankel方阵。该矩阵是一个对称矩阵，它的元素为$h_{ij}=x_{i+j-a}$。第1列为向量**x**，反三角以下的元素为0。
hankel(x,y)	返回一个$m \times n$的Hankel矩阵，它的第1列为向量**x**，最后一行为向量**y**。
magic(n)	给出一个$n \times n$的魔方矩阵。
pascal(n)	返回一个$n \times n$的Pascal矩阵，它是对称、正定的矩阵，它的元素由Pascal三角组成。它的逆矩阵的所有元素是整数。



`pascal(n, k)`	k=1时给出一个由下三角的Cholesky系数组成的Pascal矩阵。要注意的是通常情况下在 MATLAB中使用上三角的 Cholesky系数。k=2时是pascal(n,1)的变换和重新排列的形式。
`rosser`	给出Rosser矩阵，这是一个经典对称特征测试问题，它的大小是8×8。
`vander(x)`	返回一个倒数第 2列为向量 **x**的Vandermonde矩阵。它的元素 $v_{i,j}=x_i^{n-j}$，n为向量**x**的长度值。
`wilkinson(n)`	返回一个$m\times n$的Wilkinson特征值测试矩阵。

■ **例**4.13

$m\times n$的魔方矩阵由1～n^2的整数作为其元素，并且矩阵任一行和列的元素之和相等。要创建一个3×3的魔方矩阵，可以输入`magic(3)`，得到的结果为：

```
ans =
     8     1     6
     3     5     7
     4     9     2
```

■

■ **例**4.14

为了能够得到一个Householder矩阵，可以运行命令`help private/house`这将给出指定参数的信息。下面的语句可用来创建一个 Householder矩阵**H**。

```
x = [2; 5; 3];
[V, BETA] = gallery('house',x);
H = eye(3,3) - BETA*V*V'

H =
   -0.3244   -0.8111   -0.4867
   -0.8111    0.5033   -0.2980
   -0.4867   -0.2980    0.8212
```

■

表4-1中通过命令`gallery`列出了最常用的矩阵族。

表4-1　**通过命令**gallery**得到的可用矩阵族**

`circul`	给出循环序列矩阵，第1列作为参数给出，然后用相同的循环值建立矩阵的其他列
`dorr`	给出一个条件数差的对角三角矩阵
`house`	给出一个Householder矩阵
`invhess`	给出一个上Hessenberg矩阵的逆矩阵
`jordblock`	给出一个Jordan矩阵
`poisson`	给出一个稀疏对角三角块矩阵，在求解带有有限差分的 Poisson方程时要用到该矩阵
`vander`	给出一个Vandermonde矩阵
`wilk`	给出一个Wilkinson矩阵



第5章　字符串和其他数据类型

在MATLAB中可能会遇到对字符和字符串的操作。字符串能够显示在屏幕上，也可以用来构成一些命令，这些命令在其他的命令中用于求值或者被执行。细胞矩阵或者细胞数组是无类型矩阵，它们中的元素可以是任何类型。 MATLAB中还有大量适合于位运算的函数和一些常用的整数函数。还有可能把向量作为集合来看待。

5.1　字符串

一个字符串是存储在一个行向量中的文本，这个行向量中的每一个元素代表一个字符。实际上，元素中存放的是字符的内部代码，也就是ASCII码。当在屏幕上显示字符变量的值时，显示出来的是文本，而不是ASCII数字。由于字符串是以向量的形式来存储的，所以可以通过它的下标对字符串中的任何一个元素进行访问。

字符矩阵也可以这样，但是它的每行字符数必须相同。

5.1.1　分配

MATLAB中的字符串用单引号来定义：

```
NameOfVariable='text'
```

这里的'text'可以是字母、数字和特殊字符。

■ **例5.1**

(a) 简单的分配方法，如name='John Smith'，在屏幕上就会有如下显示：

```
name =
John Smith
```

(b)如果刚输入'John Smith'，那么就将这个字符串赋给变量*ans*：

```
ans =
John Smith
```

(c) 分配一个字符。如果(a)中变量**name**已存在，令name(3)='a'，则会给出：

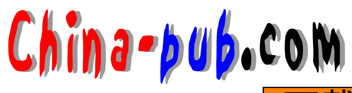
```
name =
Joan Smith
```

(d) 将上例中的字符串**name**的元素前后互换位置，可以输入：

```
for i = length(name):-1:1
  eman(i) = name(length(name)+1-i);
end
eman
```

下面显示出字符串**eman**的值：

```
eman =
htimS naoJ
```



关于循环for-loops语句可参考12.2节。用eman=fliplr(name)也可得到同样的结果；见4.1节。

(e) 一个字符串的长度：namelen=size(name)，给出：

```
namelen =
      1     10
```

(f) 在字符串中用两个单引号来表示一个单引号：

```
whoscat='Joan"s cat'
```

显示结果为：

```
whoscat=
   Joan's cat
```

(g) 字符串的组成可以象数字矩阵一样：

```
name1 = 'Joan'; name2 = 'John'; heart = 'is in love with';...
sentence = [name1,' ',heart,' ',name2]
```

显示的结果为：

```
sentence =
Joan is in love with John
```

也可参见命令集47中的命令strcat和strvcat。

(h) 冒号表达式的使用和在数字矩阵中的使用情况一样：

```
name='Charles Johnson'; firstname= name(1:7) 给出：
firstname=
Charles
```

(i) text1='John'；text2='Joan'；couple=[text1; text2] 给出：

```
couple =
John
Joan
```

■

5.1.2 字符串命令

有一些命令可用于把字符串转换成其他的表示形式。

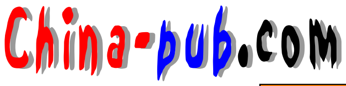
命令集45 转换字符串

abs(str)	返回一个向量，其元素是字符串**str**中字符的ASCII码值。
char(x)	根据指定的字符集将向量 **x**中的整数转换成字符。这个命令是命令abs的逆操作。在旧版MATLAB中命令setstr还可用，但将会被去掉。
num2str(f)	将数值*f*转换成浮点格式的字符串。如果需要，可包含四位数字和指数。这个命令经常和命令 disp、xlabel还有一些其他输出命令一起使用；见 13.3节例13.9。
num2str(f,k)	将数值*f*转换成带有*k*位数字的浮点格式的字符串。
num2str(f,format)	将数值*f*转换成由**format**设定格式的字符串，**format**用

	在函数sprintf中，见下面的命令。
int2str(n)	将整数*n*转换成整数字符串表达式。
rats(x,strlen)	将浮点小数*x*转换成含有对*x*的有理逼近的字符串，整数*strlen*是每个元素的字符串长度，缺省值为13。
hex2num(hstr)	将字符串**hstr**中的十六进制数转换成相应浮点数(IEEE双精度)。
hex2dec(hstr)	将字符串**hstr**中的十六进制数转换成相应整数。
dec2hex(n)	将整数*n*转换成相应的十六进制数字符串。
base2dec(str,base)	将字符串**str**中从基数*base*开始的元素转换成十进制数。
dec2base(n,base)	将整数*n*转换成基数*base*。
bin2dec(str)	将字符串中的二进制数转换成十进制数。
dec2bin(n)	将整数*n*转换成二进制数。
mat2str(A,n)	将矩阵**A**转换成字符串。如果给出了*n*值，它就代表正确数字的个数。
str2num(str)	返回字符串**str**的数字形式，字符串可以包含数字、小数点，开始的符号表示10的幂的*e*，还有复数虚部的*i*。
str2rng(str)	将一个电子表区域**str**转换成[*R*1 *C*1 *R*2 *C*2]形式的向量，它给出了字符串**str**中指定区域的开始和最后的行和列来规定电子表中的区域。
strjust(str)	返回一个和**str**一样内容的字符串，但是调整了右边。
sprintf(formatstr,A)	将矩阵**A**中元素返回到一个格式化字符串中，这个格式是由格式字符串**formatstr**来定义的，和C语言中的格式控制相似。这个命令和命令fprintf的作用是一样的，只是返回的结果是一个字符串而不是一个文件，见15.4节。
[Str,E]= sprintf(......)	返回一个上述的字符串和矩阵**E**。如果发生错误，**E**中就会得到一个错误信息字符串；如果变换正确，就返回空矩阵**E**。
sscanf(str, formatstr,mn)	返回一个矩阵，它是根据字符串**formatstr**从字符串**str**中读出数据而构成的矩阵。读出元素的最大个数为*mn*，但是这个参数是可选的。这个命令和命令 **fscanf**的作用是一样的，区别在于前者是对一个字符串而不是对文件进行操作，见15.4节。
[A, nm, E, next]= sscanf(str, formatstr,mn)	得到一个象通过命令sscanf返回的一样的矩阵**A**，也返回正确转换的元素*mn*的个数以及矩阵**E**中的错误的个数。当所有的元素都没有读取时，通过数值 *next*来指定下个元素。

■ **例5.2**

假设这些变量已被定义为：`str='ABC'；float=1.25；`



(a) 执行命令x=abs(str)，将返回：

```
x =
    65    66    67
```

这是字符A、B、C的ASCII码。

(b) 如果输入number=hex2dec(str)，MATLAB将会给出：

```
number =
        2748
```

(c) 命令：

```
numstr = num2str(float)
disp(['Number as a string = ',numstr,'!']);
```

结果为：

```
numstr =
1.25

Number as a string = 1.25!
```

为了显示变量 **numstr**，实际上它是一个字符串，可以输入 char=numstr(4)，MATLAB就会给出：

```
char=
5
```

通过命令ischar(numstr)和whos，也可知道**numstr**是一个字符串。

(d) 命令numinfo=sprintf('The number%5.2e', float)将给出：

```
numinfo =
The number = 1.25e+00
```

(e) 命令rational=rats(0.979796)将会给出一个包含浮点小数0.979796有理逼近的字符串：

```
rational =
   4995/5098
```

在命令littleRat=rats(0.979796,5)中的可选数字5将严格限定字符串的长度为5。

```
littleRat =
 48/49
```

■

命令rat在本书2.4节中定义。

■ **例**5.3

假设矩阵A被定义为：

$$\mathbf{A}=\begin{pmatrix}1 & 1 & 3\\ 2 & 4 & 7\end{pmatrix}$$

命令mat2str(A)给出：

```
ans =
[1 1 3;2 4 7]
```

假设x=4.12345，命令mat2str(A)给出：

```
ans =
4.12345

num2str(x,2)

ans =
4.1
```

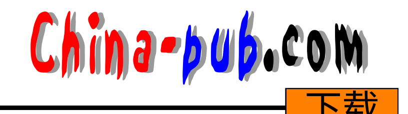
还有对字符串操作的逻辑函数和生成子串的函数。在下面的命令集46中假设str是一个字符串。

命令集46 字符串函数(一)

`blanks(n)`	返回有*n*个空格的字符串。
`deblank(str)`	返回没有后续空格的字符串 **str**。
`lower(str)`	将**str**中所有字母转换为小写字母。
`upper(str)`	将**str**中所有字母转换为大写字母。
`ischar(s)`	如果*s*是字符数据类型，则返回 1；否则返回 0。在旧版中该命令还可以使用，但是在将来的高版本中将会被去掉。
`isletter(str(i))`	如果**str**中的第*i*个字符是字母，则返回1。
`isspace(str)`	返回一个和 **str**大小相同的向量。如果在 **str**中的字符是空格、制表符或者换行符，则向量的相应位置的元素为1；否则为0。
`strcmp(str1,str2)`	比较串**str1**和串**str2**，如果相等返回1；否则返回0。
`stricmp(str1,str2)`	和**strcmp**一样，但是在比较时不区分大小写。
`str2mat(str1,str2,...)`	用**str1**、**str2**等创建字符串矩阵。如果字符串str_i的大小不同，MATLAB自动在较短的字符串后添加空格。函数最多可以带11个参数，但是它们本身也可以是字符串矩阵。
`findstr(str1,str2)`	返回一个向量，它包含 **str1**中子串**str2**的起始位置。
`strrep(str1,str2,str3)`	在字符串**str1**中含有**str2**的所有位置用**str3**来代替。
`strtok(str1,str2)`	返回在**str1**中含有**str2**的第一个标记前所有的 **str1**的部分。如果str2没有指定， MATLAB使用空格，那么就找出**str1**中不含有空格的第一个序列。
`[outstr,rstr]=strtok`	字符串outstr等于`strtok(str1,str2)`返回值，字符串**rstr**
`(str1,str2)`	等于字符串**str1**余下的部分。
`lasterr`	返回上一个错误信息的字符串。
`lastwarn`	返回上一个警告信息的字符串。

■ **例**5.4

(a) `name=upper('matlab')`给出：




```
name=
MATLAB
```

(b) fun=strrep(‘hahaha’,’a’,’i’)

```
fun=
hihihi
```

(c) 假设字符串变量定义为：

```
greet = 'Welcome', where = 'to Joan''s', party = ...
'birthday party!'
```

那么执行命令str2mat(greet,where,party)的结果为：

```
ans =
Welcome
to Joan's
birthday party!
```

(d) 可使用命令strtok从用逗号分隔内容的字符串中抽取出信息来。字符串 **text**定义为：

```
text = ...
'Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday';

[day,rest] = strtok(text,',')

day =
Monday
rest =
,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday
```

通过字符串rest来调用strtok可以找到下一天等：执行命令[day2,rest]=strtok(rest,’,’)，结果为：

```
day2 =
Tuesday
rest =
,Wednesday,Thursday,Friday,Saturday,Sunday
```

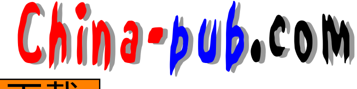
要注意的是此时调用该命令时会读取到第 2个逗号处，是因为第 1个逗号在开始位置，它并不分隔字符串的任何部分。

■

有两个用于连接字符串的命令：strcat用于连接字符串，strvcat用于把字符串连接成一个列向量。分析字符串的内容可以使用命令：strmatch和strcmp。

命令集47　字符串函数(二)

strcat(str1,str2,...)	将字符串**str1**和**str2**连接起来，它不同于对细胞矩阵操作的cat命令。
strvcat(str1,str2,...)	将**str1**和**str2**连接成一个列向量。**str1**和**str2**必须要有相同的字符个数，行数可以不相等，最后结果的总行数是它们的行数之和。
strmatch(key,strs)	检查**strs**中的各行，返回一个向量，它包含了行以字符串**key**开头的行号。



strncmp(str1,str2,n) 比较str1和str2中前n个字符，如果相等返回1；否则返回0。
strncmpi(str1,str2,n) 和命令strncmp一样，但是在比较时不区分大小写。

■ **例5.5**

(a) 假设**A**和在例5.3中定义的一样，那么：

```
text1 = 'ResultMatrix = ';
text2 = mat2str(A);
restext = strcat(text1,text2)
```

则：

```
restext =
ResultMatrix =[1 1 3; 2 4 7]
```

注意在等号和左括号之间没有空格，这是因为strcat去掉了所有的前置空格，如果想保留这些空格就应该使用命令cat。

(b) 有下列语句：

```
head = ['First name' '    ' 'Last name'];
boss = ['John' '                 ' 'Smith'];
workers = ['Arthur' '             ' 'Moore ';
           'Joseph' '             ' 'Jonson';
           'Daniel' '             ' 'Smart '];
```

在每行中间的字符串包含两个tab符号，它是用来做表格对齐的。

```
Table = strvcat(head,boss,workers)
```

```
Table =
First name       Last name
John             Smith
Arthur           Moore
Joseph           Jonson
Daniel           Smart
```

对**Table**来说，利用命令whos可知：

```
whos Table

  Name       Size       Bytes      Class

  Table      5x20         200      char array

Grand total is 100 elements using 200 bytes
```

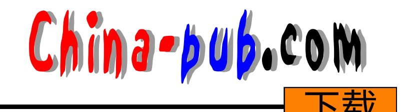
■

它的大小取决于最长的行，所以它更适合于存储这种分开形式的列表。

■ **例5.6**

语句如下：

```
carVocabulary = strvcat('car','carpool','police car')
```




```
carVocabulary =
car
carpool
police car
strmatch('car',carVocabulary)

ans =
     1
     2
```

这个结果是字符串中以'cat'开头的行号。

■

5.1.3 显示和输入

可以通过简单地输入变量的名字来显示数字矩阵或者字符串向量的内容，见 2.3节。结果将显示出变量的名字和内容。

另一种显示变量的值就是使用命令 disp。使用它只显示出变量的内容，这是有用的，特别是在字符串的应用中。

命令集48 显示命令

disp(A)	显示矩阵**A**的内容，如果**A**是字符串，则显示出它的文本。

通过input命令来接收从终端输入的内容，它也可以显示文本和提示，见附录 A.6中的例子。从例13.9可知命令disp可以和num2str和int2str结合使用。

命令集49 输入命令

input(out,*in*)	在屏幕上显示出字符串**out**的文本并等待终端的输入。如果变量**in**是's'，则输入的内容以字符串的形式进行保存，通常MATLAB在保存前要尽可能地求出表达式的值。如果使用格式控制符号如'\n'，字符串**out**可以是若干行。

■ **例**5.7

(a) 读入一个实数*x*：

```
x=input('Give a number x:')
```

显示的结果为：

```
Give a number x: 2.0944
x =
     2.0944
```

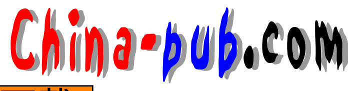
(b) 读入矩阵**A**。input命令之后的分号用来限定输出结果。

```
A=input('Give the matrix A row by row:');
```

显示并输入：

```
Give the matrix A row by row: [1  2 ; 3  5]
```

(c) 输入时还允许使用数学表达式和函数：


下载


```
A=input('Please give me a matrix: ');
```

显示并输入：

```
Please give me a matrix: rand(4)+hilb(4)
```

(d) 输入命令还可带多个参数：

```
[m, n]=input('Give the size of A:')
```

显示并输入：

```
Give the size of A: size(A)
```

(e) 逻辑表达式的计算：

```
s = input('MATLAB input\nWrite an expression: ')
MATLAB input
Write an expression: log2(8) + 7 < 10
s =
     0
```

(f) 读入一个字符串：

```
name=input('What is your last name? ');
```

显示并输入：

```
What is your last name? 'Smith'
```

注意，在这种情况下输入的字符串要用' '引出。如果命令是这样：

```
strname=input('What is your last name? ', 's');
```

就可以直接输入字符串：

```
What is your last name? Smith
```

■

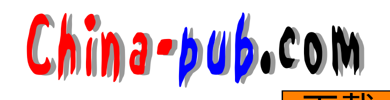
MATLAB还有一些其他的方式来读入字符串，比如通过菜单；见 14.3节。

5.1.4　字符串求值

MATLAB命令可以以字符串的形式进行输入和存储。这些命令字符串通过eval命令来求值。

命令集50　字符串求值

eval(str)	执行**str**中包含的MATLAB命令并返回结果。
eval(str1, str2)	执行**str1**中的MATLAB命令，如果没有错误就和执行eval(str1)一样；如果在对**str1**求值中第一个字符串是一个错误，则对字符串**str2**进行求值，给出一个错误信息或者其他内容。
[x1,x2,...]=evalin(aa, str)	和eval一样，决定在什么样的工作区中如读/写变量来对**str**求值。**aa**字符串可以是'caller'(在evalin调用的工作内进行求值)，也可以是'base'(在MATLAB命令执行区来求值)。结果数据存储在**x1**，**x2**, …中。
evalin(aa, str, alt)	在工作区**aa**(见上)中试着对字符串**str**求值。如果不成功，则对字符串**alt**求值。
assignin(aa, name,	把值**val**赋给变量**name**，如果变量不存在，则新建。参



val)	量aa表示工作区，见命令evalin。
g=inline(str,*arg1*, *arg2*,···)	从字符串**str**中建立一个叫内联的函数**g**，如存储在工作内存中的函数，可以用g(**val** 1,**val** 2···)来调用。函数中参数的名字可以在字符串**arg**1，**arg**2，···中给出，如果没有给出，MATLAB将从str中找出小写字母作为参数的名字。
g=inline(str,n)	用*n*+1个参量*x*，*P1*，*P2*，···，*Pn*来建立一个内联函数。
argnames(g)	用内联函数**g**中的参量名字创建一个细胞矩阵。
vectorize(g)	为了满足元素操作，在 *、/和^前加一个点‘.’，建立一个向量化的内联函数**g**。
formula(g)	把内联函数**g**作为字符串返回。
char(g)	等同于formula(g)。

命令eval使得MATLAB成为一个可灵活编程的语言。比如这个命令用来调用 MATLAB中没有预定义的函数，这很有用处，见12.4节。

■ 例5.8

(a) 假设要编写一段简单的矩阵计算的程序来输入矩阵 **A**和**B**，并且可以让用户来决定是否进行加法或者乘法操作。程序代码如下：

```
disp('Matrix analysis program. Give two matrices:');
A = input('A = ');
B = input('B = ');

choice = input('Choose one: 1 = A+B, 2 = A*B: ');

switch choice
  case 1, eval('disp(''Addition: ''); A+B');
  case 2, eval('disp(''Multiplication: ''); A*B');
end;
```

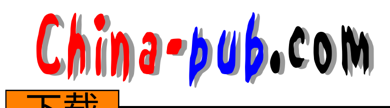
为了不结束字符串，这里应该用两个分号。在 5.1.3节对命令disp和input进行了描述，命令switch-case可参见12.1节。

由于矩阵维数的不正确等原因，如矩阵 **AB**相乘，可能会导致一些错误，所以要求矩阵的维数要一致。用第2个字符串作为命令eval的参数可以得到一个错误信息。通过运行下列程序代码可以得到：

```
disp('Matrix analysis program. Give two matrices:');
A = input('A = ');
B = input('B = ');

choice = input('Choose one: 1 = A+B, 2 = A*B :');

switch choice
  case 1, eval('disp(''Addition: ''); A+B','catchInfo');
  case 2, eval('disp(''Multiplication: ''); A*B', 'catchInfo');
end;
```



用命令eval来调用保存在文件**catchInfo.m**中的用户自定义的函数**catchInfo**。

```
function catchInfo

errStr = lasterr;
dimStr = findstr(errStr,'dimensions must agree');

if ~isempty(dimStr)              % If wrong dimensions...

  if ~isempty( findstr(errStr,'Inner'))
    disp('Error!  A*B requires A:m*n, B:n*p.');
  else
    disp('Error! Addition requires A:m*n, B:m*n.');
  end
end
```

字符串变量**lasterr**包含了MATLAB中最新的错误信息，假设给出矩阵：

```
A = [1 2 3; 4 5 6; 7 8 9], B = [1 2; 3 4].
```

相加可得：

```
Addition:
Error! Addition requires A:m*n, B:m*n.
```

相乘可得：

```
Multiplication:
Error!  A*B requires A:m*n, B:n*p.
```

(b) 字符串变量可以和字符串常量结合使用。比如有一个变量名为 file='myfile.mat'，要想将当前的变量值保存到这个文件中，可以输入 eval['save',file]，这和输入save myfile.mat是一样的。在2.8节中定义了save命令。

(c) MATLAB还可以对向量函数进行求值。令**str1**='b.*sin(k.*x)'。**b**，**k**，**x**都是向量：

```
b = [1 2 3]; k = [2 2 2]; x = [1.2 1.5 1.2];
```

执行命令：

```
values=eval(str1)
```

结果为：

```
values =
    0.6755      0.2822      2.0264
```

(d) 创建一个字符串：

```
fcn='input(''Give a function","$")'
```

在5.1.3节中提到了input命令，现在可以对从终端选择的函数进行求值操作。执行命令：

```
fplot(eval(fcn), [0,4])
```

在屏幕上显示如下结果：

```
Give a function
```

如果输入sin，MATLAB就会打开一个图形窗口显示出正弦函数。

假设一个名为**sinx2.m**的M文件(见2.9节)包含下列代码：

```
function y = sinx2(x)
y = sin(x.^2);
```

有关M文件的更多信息可参见12.3节，如果在文本*Give a function*之后输入sinx2，就可

以画出函数 $\sin x^2$ 的图形来，如图5-1所示。

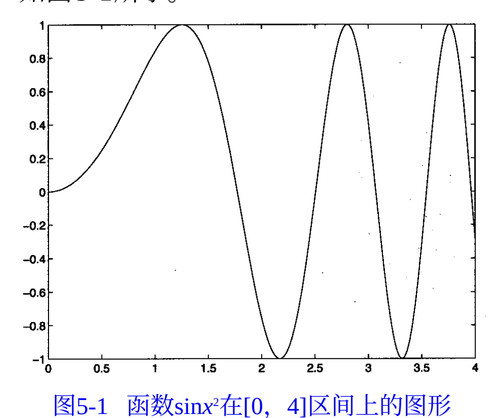


图5-1　函数 $\sin x^2$ 在[0，4]区间上的图形

■

求值的字符串也能够包含在MATLAB结构语句中，如`if`、`while`、`for`等等；见12章。

■ **例5.9**

建立一个内联函数来计算表达式 $3\sin(x)+5\cos(y)$，可以输入：

```
g = inline('3*sin(x)+5*cos(y)','x','y')

g =
     Inline function:
     g(x,y) = 3*sin(x)+5*cos(y)

argnames(g)

ans =
    'x'
    'y'

g(pi,2*pi)
ans =
     5
```

如果输入whos，可显示出：

```
  Name        Size            Bytes  Class

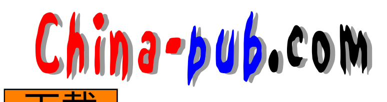
  ans         1x1                 8  double array
  g           1x1               898  inline object

Grand total is 75 elements using 906 bytes
```

■

5.2　整数

MATLAB中有大量的关于整数操作的命令。在命令集51中的 a 和 b 都是整数。



命令集51　整数函数

`mod(a,b)`	返回*a*，*b*相除后的余数。
`factor(a)`	返回*a*的素数因数。
`primes(a)`	返回一个由不大于*a*的素数组成的行向量。
`isprimes(a)`	如果*a*是一个素数，则返回1。
`nextpow2(a)`	返回最小的整数*n*，满足条件$2^n>a$。
`perms(c)`	返回向量**c**的所有可能的排列。
`nchoosek(A,k)`	返回一个每行有*k*个元素的矩阵，并列出了矩阵**A**中*k*个元素的可能每种组合。

函数perms和nchoosek随着输入返回的结果会变得很多，所以可能需要较长的时间来计算。

■ **例**5.10

判断4567是否是一个素数，执行命令`isprime(4567)`可得结果为：

```
ans=
     1
```

在4569中有哪些因数呢？可以输入命令`factor(4569)`并执行得到的结果为：

```
ans =
           3          1523
```

■

5.3　位操作

在MATLAB中可以对整数变量进行位操作，如果答案为真返回1；为假返回0。

命令集52　位操作

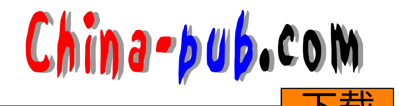
`bitand(a,b)`	返回*a*和*b*按位与操作后的值。
`bitor(a,b)`	返回*a*和*b*按位或操作后的值。
`bitxor(a,b)`	返回*a*和*b*按位异或操作后的值。
`bitget(a,bit)`	返回在整数*a*中位置*bit*的位值。
`bitset(a,bit,newbit)`	将值*newbit*赋于整数*a*中在位置*bit*的位。
`bitshift(a,n)`	对*a*进行*n*步移位操作，如果*n*为整数则左移；否则右移。
`bitcmp(a,n)`	返回*n*位*a*的补码整数。
`bitmax`	返回机器的最大浮点整数。

■ **例**5.11

令a=7，b=3，c=4，它们的二进制码为a=111，b=011，c=100。执行命令bitand(a,b)，结果为：

```
ans =
      3

bitor(b,c)
```




```
ans =
     7

bitset(a,2,0)

ans =
     5
```

■

5.4 集合

MATLAB 5中有大量的将向量和细胞矩阵解释为集合及其操作的函数。在命令集 53中变量a和b都是向量。注意这些命令都可以用添加字符串‘rows’作为最后一个参数来对矩阵操作。在这种情况下，矩阵的每一行就看成是一个单一集合，同时函数对一次一个集合作用。返回的集合中的数是有序的，并且每个数只出现一次。

命令集53 集合

intersect(a,b)	返回集合**a**和**b**的交集。
ismember(a,s)	返回一个和 **a**一样大小的 0-1向量。当 **a**的元素包含在 **s**中时，该向量中相应位置为1；否则为0。
setdiff(a,b)	返回存在集合**a**中但不存在集合**b**中的值。
setxor(a,b)	返回存在集合 **a**中但不存在集合 **b**中和存在集合 **b**中但不存在集合**a**中的所有的值。
union(a,b)	返回集合**a**和**b**的并集。
unique(a)	去掉集合**a**中重复的值，保证每个值在集合中只出现一次。

■ **例**5.12

令a=[12 24 42 24 12]b=[96 42 64]

执行命令union(a,b)，得到的结果为：

```
ans =
    12    24    42    64    96

unique(a)

ans =
    12    24    42

intersect(a,b)

ans =
    42
```

■

5.5 细胞矩阵

细胞矩阵(或细胞数组)中不同位置(细胞)可有不同数据类型，这就使得它不同于只能由数

字组成的数字矩阵和只能由字符串组成的文本矩阵。所以，一个细胞矩阵可以包含比如一个字符串、两个数字和一个细胞矩阵。细胞矩阵也可象数字矩阵一样有多维的。对于一维细胞矩阵，也类似于数字矩阵，称为细胞向量。

有三种方法来创建一个细胞矩阵。第 1种方法是使用大括号，‘ {} ’，就象用中括号‘ [] ’来创建数字矩阵一样；第 2种方法是对细胞进行逐一赋值，称为细胞赋值；第 3种方法是创建一个大小合适的空矩阵。矩阵中所有的行必须要有相同的细胞数。

■ **例**5.13

可以象以下的方法用不同数据类型直接分配：

```
A = {'John' 'Smith' 38 11.21; 'Paul' 'Anderson' 41 23.12}
```

或者是：

```
A = {'John','Smith',38,11.21; 'Paul','Anderson',41,23.12}

A =
    'John'     'Smith'        [38]     [11.2100]
    'Paul'     'Anderson'     [41]     [23.1200]
```

如果一个细胞中包含一个细胞矩阵，就要求这样来分配：

```
B = { {2 2; 1 3} 22.3; 42 21 }

B =
    {2x2 cell}      [22.3000]
    [        42]    [      21]
```

可以对某个细胞进行赋值来创建一个完整的细胞矩阵。

```
C{2,1} = 12.2

C =
               []
    [12.2000]
```

要创建一个2×4的空细胞矩阵，可以使用命令：

```
D = cell(2,4)

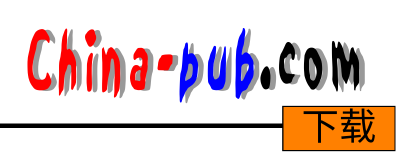
D =
     []       []       []       []
     []       []       []       []
```

■

有些函数只能作用于细胞矩阵，还有一些其他的函数如cat和strcat也可以作用于细胞矩阵。

命令集54 细胞矩阵

cell(m,n)	创建一个 $m\times n$ 的空细胞矩阵。
cell2struct(cell, posts,dim)	用细胞矩阵**cell**的元素和域**posts**创建一个结构，根据 *dim* 来决定挑选元素。*dim*=1是就是将**cell**中的第1列作为结构数1；要取出元素，*dim*应该为2。



celldisp(cell)	逐个显示cell的每个元素的值。还可以用该命令来显示 cell 中的细胞矩阵的元素。
cellplot(cell)	显示出cell的结构图。
cellstr(s)	用s中每一行作为一个细胞来创建一个细胞向量，命令 char是这个函数的逆操作。
iscellstr(cell)	如果cell中只含有字符串，则返回 1；如果cell中既有细胞矩阵又有字符串，则返回 0。
num2cell(A,dim)	返回一个和矩阵 A一样大小的细胞矩阵。如果给出了参量 *dim*，在它自己的细胞中将 *dim*维作为一个向量，这样得到的矩阵就和 A的大小不一样。
[out1,out2,···]= deal(in1,in2,···)	将输入拷贝到输出中，如 out1=in1，out2=in2等；参见helpdesk中有关细胞矩阵和结构的例程。

■ **例** 5.14

执行如下命令可以得到例5.13中细胞矩阵B的每个元素结构图：

```
cellplot(B)
```

结果如图5-2所示。

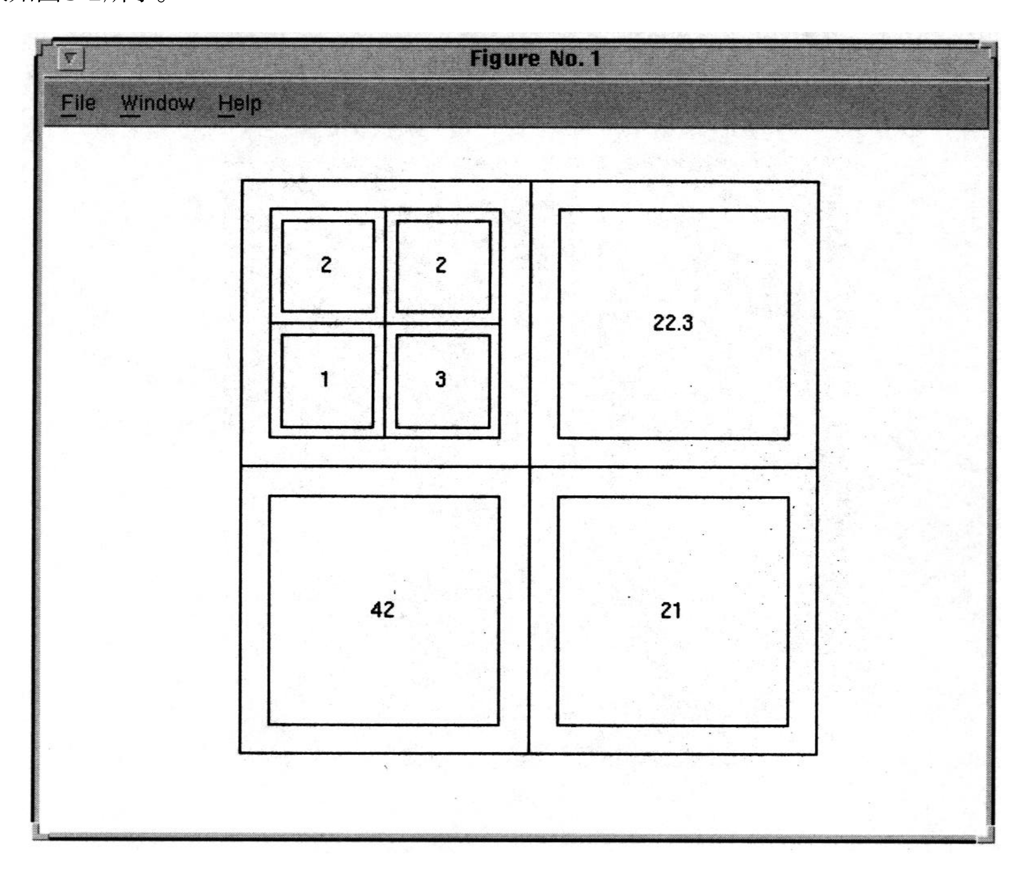
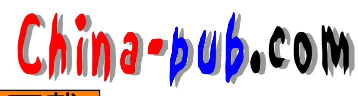


图5-2　细胞矩阵的图形表示



用命令celldisp(B)来显示细胞矩阵的每个元素的值：

```
celldisp(B)


B{1,1}{1,1} =
      2
B{1,1}{2,1} =
      1
B{1,1}{1,2} =
      2
B{1,1}{2,2} =
      3
B{2,1} =
     42
B{1,2} =
   22.3000
B{2,2} =
     21
```

■



第6章　数据分析和统计

在本章中将介绍MATLAB对数据处理和统计分析的命令。如果没有特别强调，本章中的A和B是指$m\times n\times\cdots\times p$的多维矩阵，x是一个向量。

6.1　最大值和最小值

用命令集55中列出的命令可以求解最大值。

命令集55　最大值和最小值

`max(x)`	返回**x**中最大的元素值，如果x是复数，则返回max(abs(x))值。
`max(A)`	返回一个含有**A**中第1维最大值的$1\times n\times\cdots\times p$矩阵。对于二维矩阵来说，返回一个行向量，它的第 1个元素是**A**中第1列的最大的元素；如果**A**为复数时，则返回max(abs(A))值。
`[y,ind]=max(A)`	返回一个含有**A**中第1维最大值的$1\times n\times\cdots\times p$矩阵***y***，并在行向量**ind**中保存每列的最大数的行下标。
`max(A,B)`	返回一个和**A**、**B**相同维数的矩阵，每一元素都是在 **A**和**B**中的相同位置上是最大的元素。
`C=max(A,[],dim)`	给出在指定的*dim*维内**A**的最大分量。如max(A,[],1)，则给出**A**中最大的行向量。
`min(x)`	返回向量x中最小的元素。该命令关于矩阵的操作和max一样，如果*x*是复数，则返回min(abs(x))值。

■ **例6.1**

创建一个三维矩阵A：

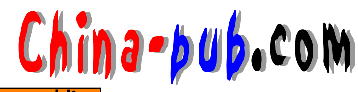
```
A(:,:,1) = [1 2 3;2 3 1;3 2 1];
A(:,:,2) = [2 4 6;4 6 2;6 4 2];
```

显示结果为：

```
A(:,:,1) =
      1      2      3
      2      3      1
      3      2      1
A(:,:,2) =
      2      4      6
      4      6      2
      6      4      2
```

求其中的最大值可用命令：

```
max(A)
```




```
ans(:,:,1) =
     3     3     3
ans(:,:,2) =
     6     6     6
```

输入whos可获得A的详细情况：

```
  Name        Size          Bytes  Class

  A           3x3x2           144  double array
  ans         1x3x2            48  double array

Grand total is 24 elements using 192 bytes
```

■

6.2　求和、乘积和差分

使用命令sum和cumsum可以求得各种不同的和。

命令集56　求和

sum(x)	返回向量x所有元素的和。
sum(A)	返回一个包含矩阵A各列元素之和的$1\times n\times\cdots\times p$矩阵。
cumsum(x)	返回一个x中元素累计和的向量，也就是第 2个元素是x中前两个元素之和，以此类推。
cumsum(A)	返回一个与A同样大小的矩阵，它的列是A中列的累计和。
cumsum(A,dim)	给出A中*dim*维的元素累计和，命令 cumsum(A)和命令cumsum(A,1)相同。

■ **例6.2**

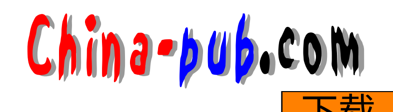
对例6.1中的矩阵A进行求和及累计和，其操作如下：

```
TheSum = sum(A), TheCsum = cumsum(A)

TheSum(:,:,1) =
     6     7     5
TheSum(:,:,2) =
    12    14    10

TheCsum(:,:,1) =
     1     2     3
     3     5     4
     6     7     5
TheCsum(:,:,2) =
     2     4     6
     6    10     8
    12    14    10
```

图6-1说明了MATLAB是如何来计算sum(A)的。各列相加后存放到$1\times3\times2$的三维数组TheSum中。



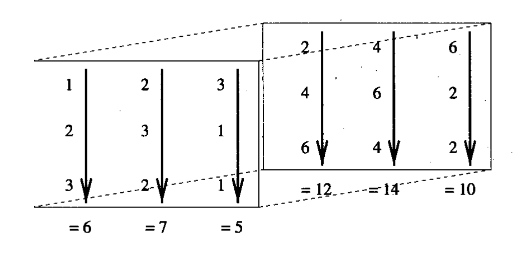


图6-1 三维数组中求和 sum(A) 示意图

■

乘积的计算也和这相似。

命令集57 乘积

prod(x)	返回**x**中各元素乘积。
prod(A)	返回一个元素是列乘积的多维矩阵。
prod(A,dim)	给出*dim*维内的元素乘积。
cumprod(x)	返回一个**x**中各元素累计积的向量，也就是第 2个元素是**x**中前两个元素的累计积，以此类推。
cumprod(A)	返回一个矩阵，其中列元素是**A**中列元素的累计积。
cumprod(A, dim)	给出在*dim*维内的累计积。

■ **例** 6.3

假设使用定义在例6.1中的矩阵**A**，执行命令：

```
TheProd = prod(A), TheCprod = cumprod(A)
```

返回得到：

```
TheProd(:,:,1) =
     6    12     3
TheProd(:,:,2) =
    48    96    24

TheCprod(:,:,1) =
     1     2     3
     2     6     3
     6    12     3
TheCprod(:,:,2) =
     2     4     6
     8    24    12
    48    96    24
```

■

使用命令diff可以进行差分计算，还有一些与命令 diff相关的其他命令。

命令集58 差分和梯度

diff(x)	给出一个长度为$n-1$的向量，它的元素是长度为*n*的向量**x**中相邻的

	元素的差。如果**x**=(x_1 x_2 … x_n)，则diff(**x**)=(x_2－x_1 x_3－x_2…x_n－x_{n-1})。
`diff(A)`	在**A**的第一维内计算相邻元素的差分。对于二维矩阵来说，就是diff(A)=A(2:m,:)－A(1:m－1,:)。
`diff(x,k)`	求出第*k*次差分，diff(x,2)和diff(diff(x))等价。
`diff(A,k,dim)`	在*dim*维内求出第k次差分。
`[DAdx,DAdy, DAdz,...]= gradient(A)`	在矩阵**DAdx**、**DAdy**、**DAdz**等中返回矩阵**A**的偏导数，每个矩阵包含$\partial A/\partial x$、$\partial A/\partial y$、$\partial A/\partial z$等相应的下标。在MATLAB中输入help gradient可得到更多信息，也可参见例13.16。
`[DAdx,DAdy, DAdz,...]= gradient(A,h1, h2,h3,...)`	返回偏导数$\partial A/\partial x$、$\partial A/\partial y$、$\partial A/\partial z$等，如果给出参量*h*1，*h*2，*h*3…，可将它们用作每个变量的步长。
`del2(A)`	返回离散拉普拉斯算子，矩阵中的元素为 **A**中元素和它相邻的四个元素的平均值的差分。

为了使函数$z=f(x,y)$的梯度的四个极值形象化，在颜色盘一节的图P-5中使用了gradient命令。

■ **例**6.4

差分计算很容易，它还可以当作导数的近似值来用。

```
x = [1 4 9 16 25];
d1 = diff(x), d2 = diff(d1), d3 = diff(d2)
```

得到的结果为：

```
d1 =
     3     5     7     9

d2 =
     2     2     2

d3 =
     0     0
```

■

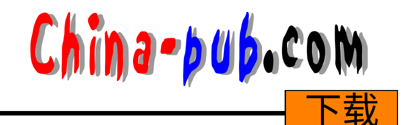
注意，如果将计算得到的差分作为导数的近似值来用，必须除以两点之间的距离。

6.3 统计命令

在前一节中提到了对矩阵列操作的命令，比如 max、min、sum和prod。下面给出了数据统计分析的命令描述。

命令集59 平均值、中值和标准差

`mean(x)`	求出向量**x**的算术平均值。



`mean(A,dim)`	给出一个 $1\times n\times\cdots\times p$ 的矩阵，它包含 **A** 中第1维的各个平均值。如果给出了 *dim*，就在 *dim* 维内计算。
`median(x)`	求出向量 **x** 中元素的中值。
`median(A,dim)`	给出一个 $1\times n\times\cdots\times p$ 的矩阵，它包含 **A** 中第1维各列的中值。如果给出了 *dim*，就在 *dim* 维内计算。
`std(x)`	求出向量 **x** 中元素的标准差。
`std(A,dim)`	给出一个 $1\times n\times\cdots\times p$ 的矩阵，它包含 **A** 中第1维的各列标准差。如果给出了 *dim*，就在 *dim* 维内计算标准差。

■ **例6.5**

令A为：

$$\mathbf{A}=\begin{pmatrix}1 & 1\\ 2 & 2\\ 3 & 3\\ 4 & 100\end{pmatrix}$$

然后执行下列命令：

```
average = mean(A), med = median(A), dev = std(A)
```

返回得到：

```
average =
    2.5000   26.5000

med =
    2.5000    2.5000

dev =
    1.2910   49.0068
```

MATLAB中命令cov和corrcofe是用来求协方差和相关系数的，这些命令只能用在二维矩阵中。

■

命令集60　协方差和相关系数

`cov(x)`	求向量 **x** 的协方差。
`cov(A)`	求协方差矩阵，对角线元素是 **A** 中各列的方差。
`cov(x,y)`	等同于 `cov([x y])`，**x** 和 **y** 是列向量。
`corrcoef(A)`	求相关矩阵。
`corrcoef(x,y)`	等同于 `corrcoef([x y])`，**x** 和 **y** 是列向量。

■ **例6.6**

假设定义如下向量：

$$\mathbf{x}=\begin{pmatrix}1\\1\\1\end{pmatrix}\qquad \mathbf{y}=\begin{pmatrix}1\\2\\2\end{pmatrix}\qquad \mathbf{z}=\begin{pmatrix}0\\-1\\1\end{pmatrix}$$

(a) 通过下列命令来求得方差：

```
varx = cov(x), vary = cov(y), varz = cov(z)
varx =
     0

vary =
    0.3333

varz =
     1
```

(b) 协方差为：

```
Cvxy = cov(x,y), Cvxz = cov(x,z), Cvyz = cov(y,z)
Cvxy =
         0         0
         0    0.3333

Cvxz =
     0     0
     0     1

Cvyz =
    0.3333         0
         0    1.0000
```

(c) 通过下列命令来求得相关矩阵：

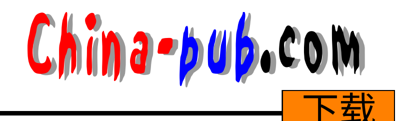
```
Corrxy = corrcoef(x,y), Corrxz = corrcoef(x,z), ...
Corryz = corrcoef(y,z)

Warning: Divide by zero.
> In /opt/matlab52/toolbox/matlab/datafun/corrcoef.m at line 31
Corrxy =
   NaN   NaN
   NaN     1

Warning: Divide by zero.
> In /opt/matlab52/toolbox/matlab/datafun/corrcoef.m at line 31
Corrxz =
   NaN   NaN
   NaN     1

Corryz =
     1     0
     0     1
```

■



6.4 排序

在MATLAB中可以用命令sort来进行数据排序。

命令集61 排序

`sort(x)`	返回一个向量**x**的元素按递增排序的向量。如果元素是复数，则使用绝对值进行排序，即sort(abs(x))。
`[y,ind]=sort(x)`	返回下标向量**ind**。就是y=x(ind)。另外向量**y**是**x**中元素按递增排序得到的。
`sort(A,dim)`	对**A**中各列按递增排序，注意矩阵的行已被改变。如果给出了*dim*，则在*dim*维内进行排序。
`[B, Ind]=sort(A)`	返回矩阵**Ind**和矩阵**B**，矩阵**B**的列为矩阵**A**中按递增排序的列，矩阵**Ind**的每列相对应于上面提到的向量中列**ind**。
`sortrows(X, col)`	对矩阵**A**的各行按递增排序。如果行的元素是复数，它们以abs(x)为主，以angle(x)为辅进行排序。如果给出*col*，则根据指定的列数对行进行排序。

■ **例6.7**

假设矩阵**A**为：

$$\mathbf{A}=\begin{pmatrix}0&4&4\\2&0&2\\4&2&0\end{pmatrix}$$

(a) 执行命令[Ascend,Ind]=sort(A)，结果为：

```
Ascend =
      0       0       0
      2       2       2
      4       4       4

Ind =
      1       2       3
      2       3       2
      3       1       1
```

(b) 如果以递减排序，可以使用下列命令：

Descend=flipud(sort(A))，执行后的结果为：

```
Descend =
      4       4       4
      2       2       2
      0       0       0
```

在4.1节定义了命令flipud。

■

6.5 统计频数直方图和棒图

使用命令hist、bar和stairs，将数据集合以统计频数直方图和棒图显示出来。



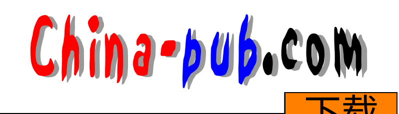
命令集62 统计频数直方图和棒图

命令	说明
`hist(x)`	在10个等分点内画出**x**中数据的统计频数直方图。
`hist(x,n)`	在*n*个等分点内画出**x**中数据的统计频数直方图。
`hist(x,y)`	在由向量y定义的等分点内画出**x**中数据的统计频数直方图，向量**y**中元素按递增排序。
`bar(x)`	画出**x**的棒图。
`bar(z,x)`	在由向量**z**定义的位置上画出**x**的棒图，**z**中的值必须是递增的均一分布。
`bar(x,...,str)`	画出如上的棒图，但可根据字符串**str**来设定颜色和形状。有关**str**的值，可参见13.1节。
`bar(A)`	画出以行分组的二维矩阵**A**的棒图。
`stairs(x)`	画出阶梯图，也就是内部没有线条的棒图。
`stairs(z,x)`	在由向量**z**定义的位置上画出储存在**x**中数据的棒图。
`barh(x,A,`*`format`*`)`	把$m \times n$矩阵**A**画成*m*组*n*个棒图，可以通过字符串**format**来指定颜色，参见13.1节；或者用字符串'`stacked`'表示把同一列的数据画在一个直方条上。
`barh(A)`	和`barh`一样，但是`x=1:m`。
`stem(y)`	在x轴上画y的离散火柴杆图，以小圆代表数据，并以其为结束。
`stem(z,y)`	在**x**轴上由向量**x**指定的位置上画y的离散火柴杆图，以小圆代表数据，并以其为结束。
`pareto(y,`*`x`*`)`	按递减顺序画出向量**y**的棒图。给出的向量**x**可作为**x**轴的坐标。如果**x**坐标没有给出，则可用向量**y**中元素的下标。该命令也是用元素的累加和来画直线。
`pie(x,`*`extract`*`)`	画向量**x**的饼图。如果`sum(x)<=1`，则画出的是一个不完整的饼图，向量**extract**的大小和**x**一样；从x中取出与**extract**中每个非零的元素相同的元素。

命令`hist`、`bar`和`stairs`还可以用来在向量中存储数据。这些命令中的一些命令如`bar`和`pie`还可作用于三维数组。

命令集63 图表

命令	说明
`[m,y]=hist(x)`	在**x**的最大值和最小值之间等分成10个区间，在这个区间上画出统计频数直方图。向量**y**的元素为将`min(x)`和`max(x)`之间分成10个等间距的值，向量**m**为在每个区间内值的个数。这个直方图也可用命令bar(y,m,'.') 来画。
`[m,y]=hist(x,n)`	在*n*个等间距区间上画统计频数直方图。
`[m,y]=hist(x,y)`	在由向量**y**指定的区间上画统计频数直方图。
`[xb,yb]=bar(y)`	画**y**的棒图，这个棒图可以用命令`plot(xb,yb)`来画。
`[xb,yb]=bar(x,y)`	在由向量**x**指定的位置上画**y**的棒图
`[xb,yb]=stairs(y)`	画**y**的阶梯图。
`[xb,yb]=stairs(x,y)`	在向量**x**定义的区间上画**y**的阶梯图。



■ **例**6.8

假设x为：

```
x = [1 1 3 4 5 1 9 8];
```

(a) 输入`hist(x)；title('Histogram of x using hist(x)')`
运行的结果如图6-2所示。

命令`title`可在图上添加文本标题，参见13.3节。

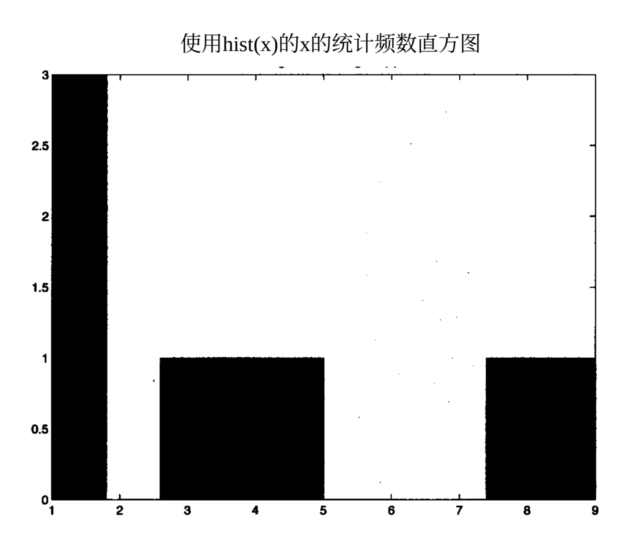


图6-2 在标准区间内的统计频数直方图

(b) 要画出在三个区间内的统计频数直方图，可输入：

```
hist(x,3); title('Histogram of x using hist(x,3)')
```

结果如图6-3所示：

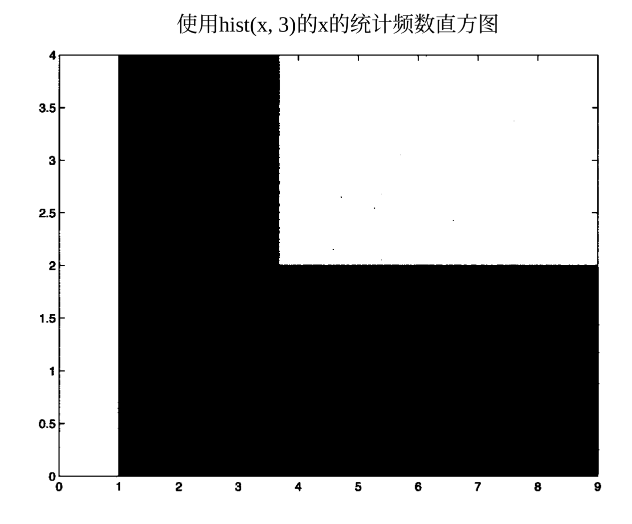


图6-3 在三个区间内的统计频数直方图

(c) 画棒图可以输入：

```
bar(x); title('bar(x)');
```

给出的图形如图6-4所示。

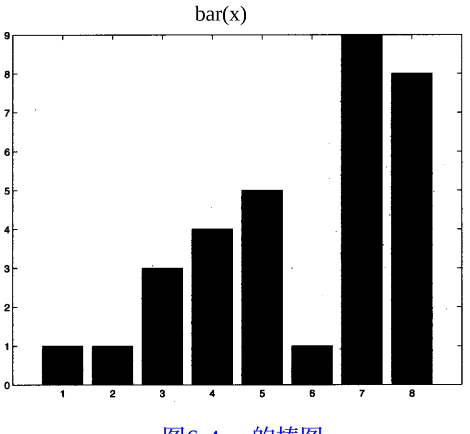


图6-4　*x*的棒图

(d) 如果输入 [m,y]=hist(x) ；MATLAB将创建出向量 **m**和**y**。如果再用命令bar(y,m,'w')，将再画出统计频数直方图；见图 6-5。为了使图形更有趣，可以画出白色的图形。用命令plot能画出其他色彩的图形来，参见13.1节。

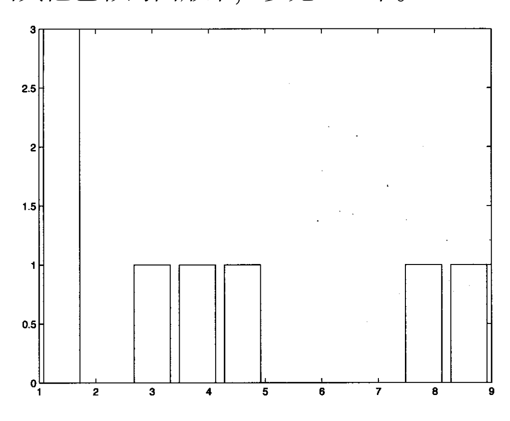


图6-5　用bar 命令画的统计频数直方图

(e) 用命令stem(x)来画出向量x的火柴杆图，结果如图6-6所示。

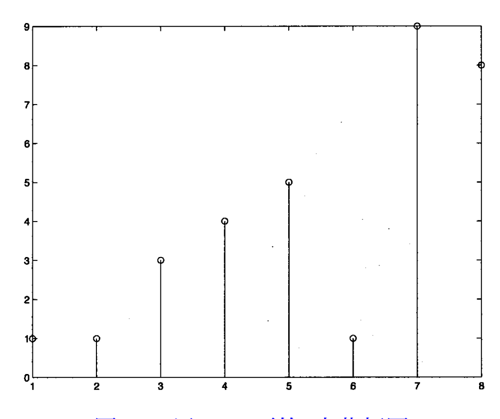
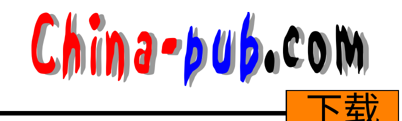


图6-6　用stem 画的**x**火柴杆图



如果再定义一个向量：

```
xvalues = [1.1 1.3 2 2.4 2.5 1.8 3 3.2];
```

用它来做x轴的坐标值，运行命令stem(xvalues,x,‘-.’)后可得到如图6-7所示的图形。

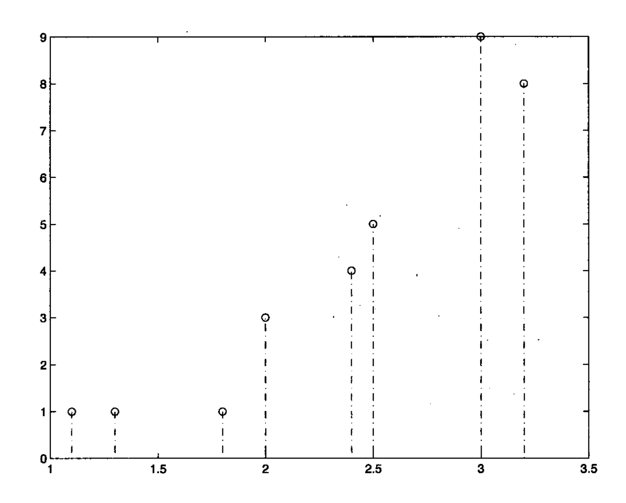


图6-7　以向量xvalues 为x轴坐标画出的x数据火柴杆图

注意，向量x的元素不必按递增排列。命令 stem的第3个参数用来确定线型，就象命令bar在画图时可指定颜色一样。

■

6.6　区域的三角分解

MATLAB中有下列关于区域三角形的命令。

命令集64　三角分解

TRI=delaunay(x,y,’sorted’)	画一个三角形将向量 x和y连接起来。如果给定参数‘*sorted*’，则假定没有重复数据，对y中的数据按递增排序。如果y的数据相同，则对x中的数据按递增排序。
voroni(x,y,TRI)	画集合x和y的Voronoi图形。如果给定 **TRI**，则画出集合x和y的Delaunay三角形图形。

有两个使用Delaunay三角形的函数，它们能给出与三角集合相关的集合信息。

命令集65　三角分解时的搜索函数

dsearch(x,y,TRI,px,py)	找到最接近(*px*, *py*)的由向量x和y定义的点的下标，矩阵**TRI**是x和y的三角矩阵。
tsearch(x,y,TRI,px,py)	找到由向量 x和y形成的集合定义的三角形下标和最接近点(*px*, *py*)的三角矩阵**TRI**，它是x和y的三角矩阵。

6.7　多边形分析

以上是两个关于多边形属性的函数。

命令集66　多边形

polyarea(x,y)	画一个由**A**和**B**的列组成的集合定义的多边形。如果给定*dim*，则画出定义在*dim*维内的多边形。
polyarea(A,B,*dim*)	画一个由**A**的第1维组成的集合定义的多边形。如果给定*dim*，则画出定义在*dim*维内的多边形。
IN=inpolygon(x,y,px,py)	返回一个和**x**，**y**大小相同的向量**IN**。如果点(x,y)在由**px**和**py**定义的多边形内，则将**IN**中相等元素赋值为1；如果点在多边形上，则赋值为0.5；在多边形外，则赋值为0。
rectint(x,y)	画由向量x和y定义的矩形。
rectint(A,B)	从rectint(A(i,:),B(j,:))中返回所有可能组合的$n\times m$矩阵，如果**A**是一个$n\times 4$的矩阵，**B**是一个$m\times 4$的矩阵。
convhull(x,y,*TRI*)	返回由**x**和**y**定义的点的下标，这个点在集合的凸起的位置上。如果给定了**TRI**，则用它；否则计算三角形。

■ **例**6.9

用矩阵**DartBoard**可以定义一个方形的镖盘。

```
DartBoard(1,:,1) = [ 2 3 3 2 2 ];
DartBoard(2,:,1) = [ 2 2 3 3 2 ];
DartBoard(1,:,2) = [ 1 4 4 1 1 ];
DartBoard(2,:,2) = [ 1 1 4 4 1 ];
```

见图6-8。

现在有：

```
DartBoard(:,:,1) =
     2     3     3     2     2
     2     2     3     3     2

DartBoard(:,:,2) =
     1     4     4     1     1
     1     1     4     4     1
```

下面的函数文件对于了解一个训练有素的镖手很有用。

```
function answer = dartresult(x,y, DartBoard)
% Returns a value corresponding to the score.

answer =
     2*sum(inpolygon(x,y,DartBoard(1,:,1),DartBoard(2,:,1)))+...
     3*sum(inpolygon(x,y,DartBoard(1,:,2),DartBoard(2,:,2)));
```

现在让MATLAB投一些镖。




```
RandomDarts(1,:) = 5.*rand(1,5);
RandomDarts(2,:) = 5.*rand(1,5)

RandomDarts =
     4.7506     1.1557     3.0342     2.4299     4.4565
     3.8105     2.2823     0.0925     4.1070     2.2235
```

可以得到一个好看的图形和有以下序列的结果，在图上把镖作为星画出来，在镖盘周围是空条纹。最后的结果放在图的上边。

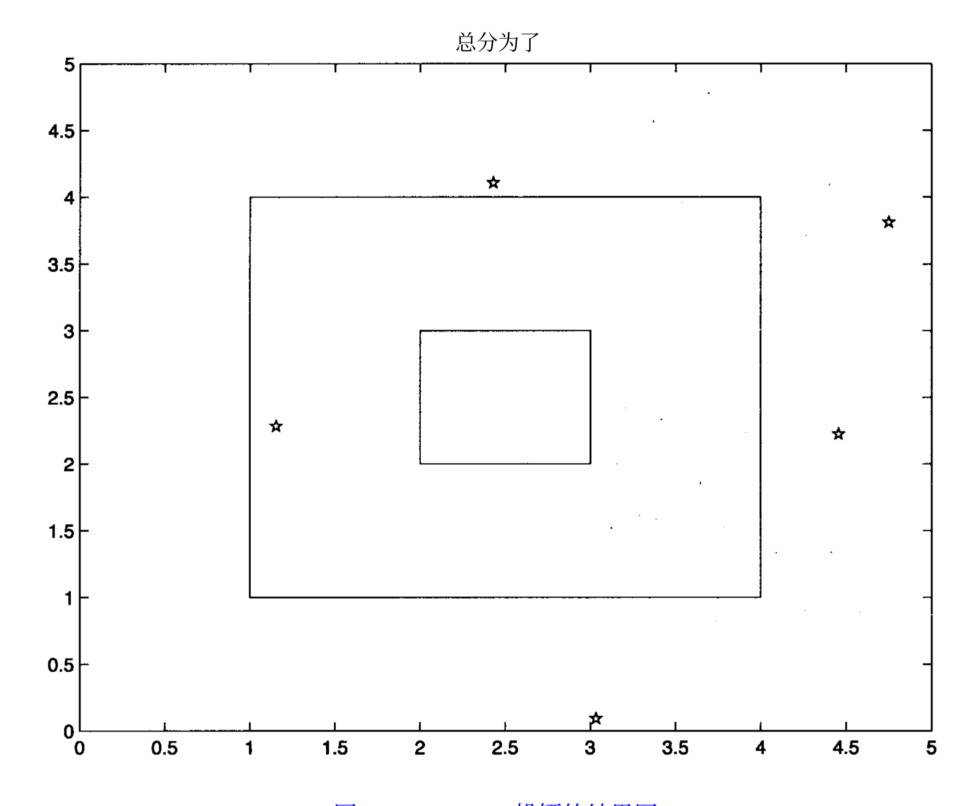


图6-8 MATLAB投镖的结果图

```
plot(DartBoard(1,:,1),DartBoard(2,:,1),'r',...
     DartBoard(1,:,2),DartBoard(2,:,2),'b')
hold on
plot(RandomDarts(1,:),RandomDarts(2,:),'pentagram')
axis([0 5 0 5])
title(cat(2,'The total score is ',num2str(...
dartresult(RandomDarts(1,:),RandomDarts(2,:),DartBoard)))
```

结果如图6-8所示。

■



第7章　线性方程系统

线性方程系统是最常见的计算问题，几乎在所有的应用中都把它作为子问题提出来。通常MATLAB用运算符\从左分开来求解线性方程系统，超定系统的解决方法同样可用于欠定系统。

在线性系统理论中重要的概念是行列式、逆和矩阵的秩。首先定义这些 MATLAB命令，再在7.2节中开始介绍求解的方法。在范数和条件数定义之后，介绍一些因数分解。最后一节涉及超定和欠定系统。

注意，本章中的所有命令只能用于二维矩阵。

7.1　行列式、逆和秩

下列命令用来计算矩阵**A**的行列式、逆和矩阵的秩。

命令集67　矩阵函数

`det(A)`	求方阵**A**的行列式。
`rank(A)`	求**A**的秩，即**A**中线性无关的行数和列数。
`inv(A)`	求方阵**A**的逆矩阵。如果 **A**是奇异矩阵或者近似奇异矩阵，则会给出一个错误信息。
`pinv(A)`	求矩阵**A**的伪逆。如果**A**是$m\times n$的矩阵，则伪逆的大小为$n\times m$。对于非奇矩阵**A**来说，有`pinv(A)=inv(A)`。
`trace(A)`	求矩阵**A**的迹，也就是对角线元素之和。

■ **例**7.1

假设有下列矩阵：

$$\mathbf{A1}=\begin{pmatrix}1 & 3\\ 2 & 4\end{pmatrix}\quad \mathbf{A2}=\begin{pmatrix}1 & 3\\ 2 & 6\end{pmatrix}\quad \mathbf{A3}=\begin{pmatrix}1 & 3 & 2\\ 2 & 7 & 6\end{pmatrix}$$

将命令`det`、`inv`、`rank`和一些其他命令对上述矩阵进行操作。

(a) `det1 = det(A1), det2 = det(A2), det3 = det(A3)`

give:

```
det1 =
    -2

det2 =
     0

??? Error using ==> det
Matrix must be square.
```

仅为方阵定义行列式。

(b) 仅为方阵定义逆：

```
Inv1 = inv(A1), Inv2 = inv(A2), Inv3 = inv(A3)
```

结果为：

```
Inv1 =
   -2.0000    1.5000
    1.0000   -0.5000

Warning: Matrix is singular to working precision.
Inv2 =
   Inf   Inf
   Inf   Inf

??? Error using ==> inv
Matrix must be square.
```

对所有矩阵定义伪逆：

```
Pinv1 = pinv(A1), Pinv2 = pinv(A2), Pinv3 = pinv(A3)
```

结果为：

```
Pinv1 =
   -2.0000    1.5000
    1.0000   -0.5000

Pinv2 =
    0.0200    0.0400
    0.0600    0.1200

Pinv3 =
    0.9048   -0.3333
    1.0476   -0.3333
   -1.5238    0.6667
```

注意，**A1**的逆矩阵和它的伪逆是一样的。

(c) 如果**A**的逆矩阵存在，那么它的行列式 $\det(A^{-1})$就等于：$\dfrac{1}{\det(A)}$

```
detinv1 = det(inv(A1))

detinv1 =
   -0.5000
```

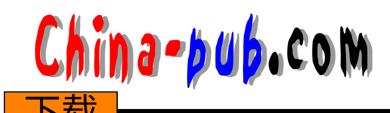
(d) 矩阵的秩和它的转置的秩相同：

```
rank1 = rank(A1), rank2 = rank(A2), rank3 = rank(A3)
```

结果为：

```
rank1 =
     2

rank2 =
     1
```




```
rankT3 =
     2
```

和

```
rankT1 = rank(A1'), rankT2 = rank(A2'), rankT3 = rank(A3')
```

结果为：

```
rankT1 =
     2

rankT2 =
     1

rankT3 =
     2
```

(e) 实数矩阵的行列式和它的转置的行列式相同：

```
detT1 = det(A1'), detT2 = det(A2'), detT3 = det(A3')

detT1 =
    -2

detT2 =
     0
??? Error using ==> det
Matrix must be square.
```

■

与线性系统相联系的两个子空间是值域和零空间。如果 **A**为$m \times n$的矩阵，它的秩为r，那么**A**的向量空间就是由**A**的列划分的线性空间，这个空间的维数是r，也就是**A**的秩。如果r=n，则**A**的列线性无关。MATLAB命令orth用来求**A**的空间的正交基。

A的零空间是由满足条件**Ax=0**的所有向量**x**组成的线性子空间。在MATLAB中可以用命令null来求得零空间的正交基。

假设有一个向量集$\mathbf{v}_1$，$\mathbf{v}_2$，…$\mathbf{v}_n$，可以通过定义矩阵$\mathbf{B}=(\mathbf{v}_1\ \ \mathbf{v}_2\ \ \cdots\ \mathbf{v}_n)$来判断它们是否线性相关。例如，如果**B**的秩是$n-1$，那么其中的一个向量v_i可以用其他向量线性表示。

用命令subspace来求得两个向量或者两个子空间的夹角。

命令集68 值域、零空间和子空间的夹角

orth(A)	求**A**空间的正交基，它的列数等于**A**的秩。
null(A)	求**A**的零空间的正交基，它的列数等于零空间的维数。
subspace(x,y)	求列向量**x**和**y**的夹角，向量的长度必须一样。
subspace(A,B)	求由矩阵**A**和**B**的列划分的子空间的夹角，列的长度必须一样。

■ 例7.2

用命令orth、null和subspace求解例7.1中的矩阵的正交基、零空间和夹角。

(a) 先求正交基：

```
Range1 = orth(A1), Range2 = orth(A2), Range3 = orth(A3)
```



结果为：

```
Range1 =
    0.5760    0.8174
    0.8174   -0.5760
Range2 =
    0.4472
    0.8944

Range3 =
    0.3667   -0.9303
    0.9303    0.3667
```

(b) 再求它们的秩：

```
rank1 = rank(orth(A1)), ...
rank2 = rank(orth(A2)), rank3 = rank(orth(A3))
```

结果为：

```
rank1 =
     2

rank2 =
     1

rank3 =
     2
```

当然，矩阵的向量空间的秩就等于矩阵本身的秩。

(c) 然后求它们的零空间：

```
nullSpace1 = null(A1), ...
nullSpace2 = null(A2), nullSpace3 = null(A3)
```

结果为：

```
nullSpace1 =
   Empty matrix: 2-by-0

nullSpace2 =
   -0.9487
    0.3162

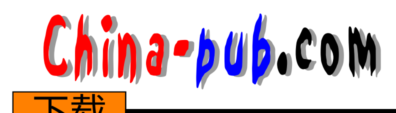
nullSpace3 =
   -0.8729
    0.4364
   -0.2182
```

这里的Empty Matrix表示零空间是空的。对**A1**求它的零空间时结果就得到零向量。

(d) 求它们转置矩阵的零空间：

```
nullSpaceT1 = null(A1'), ...
nullSpaceT2 = null(A2'), nullSpaceT3 = null(A3')
```

结果为：


下载


```
nullSpaceT1 =
   Empty matrix: 2-by-0

nullSpaceT2 =
   -0.8944
    0.4472

nullSpaceT3 =
   Empty matrix: 2-by-0
```

(e) 用orth求正交基：

```
RangeT1 = orth(A1'), ...
RangeT2 = orth(A2'), RangeT3 = orth(A3')
```

结果为：

```
RangeT1 =
    0.4046    0.9145
    0.9145   -0.4046

RangeT2 =
    0.3162
    0.9487

RangeT3 =
    0.2197   -0.4357
    0.7508   -0.4958
    0.6229    0.7513
```

(f) 最后求矩阵的夹角：

```
angle = subspace(null(A2),orth(A2'))
```

结果为：

```
angle =
    1.5708
```

■

角度为π/2，说明这两个空间是正交的。命令subspace要求列的长度相同。

7.2 线性系统的求解和LU因式分解

MATLAB中用运算符\求解线性系统，这个运算符的功能很强大，而且具有智能性。通常，仔细研究计算过程是有价值的，在MATLAB中有几个专门这样的命令。

令**A**是$n \times m$的矩阵，**b**和**x**是有n个元素的列向量，**B**和**X**是n行p列的矩阵。MATLAB用如下命令求解系统**Ax=b**：

```
x=A\b
```

求解更一般的系统**AX=B**，也用同样的方法，其中$\mathbf{B}=(\mathbf{b}_1\ \mathbf{b}_2 \cdots \mathbf{b}_p)$：

```
X=A\B
```

如果A是一个奇异矩阵，或者是近似奇异矩阵，则会给出一个错误信息。

■ **例**7.3

在MATLAB中求解下列方程组：

$$\begin{cases} 2x_1 + 3x_2 = 7 \\ 4x_1 + \ x_2 = 9 \end{cases}$$

先找出系数矩阵和右边的b：

$$\mathbf{A} = \begin{pmatrix} 2 & 3 \\ 4 & 1 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 7 \\ 9 \end{pmatrix}$$

解向量是x=(x_1 x_2)´，可以用x=A\b来求得：

```
x =
       2
       1
```

■

MATLAB依据系数矩阵 **A**的不同而相应地使用不同的方法求解线性系统。如果可能，MATLAB先分析矩阵的结构。例如，如果**A**是对称且正定的，则使用Cholesky分解。

如果没有找到可以替代的方法，则采用高斯消元法和部分主元法。主要是对矩阵进行 LU因式分解或LU分解。这种方法就是令**A=LU**，其中**U**是一个上三角矩阵，**L**是一个带有单位对角线的下三角矩阵。

然而为了保证计算的稳定性可以使用部分主元法。也就是说，**L**通常是一个改变序列的下三角矩阵，即有些行进行互换。这样，**L**就可能显得结构不完整，将这些排列定义为交换矩阵**P**。

交换矩阵P的大小为$n \times n$，它实际上是一个单位矩阵，按照交换的顺序来交换列向量。交换矩阵的逆等于它本身的转置。

$$\mathbf{PA} = \mathbf{L}_l\mathbf{U}$$

LU因式分解可以用非交换下三角矩阵L_l表示出来：

即交换矩阵**L由L=P´**$\mathbf{L}_l$给出。

MATLAB中用命令lu可以求得**U**和交换或非交换下三角矩阵**L**，后一种情况也可给出交换矩阵**P**。

命令集69 **LU分解**

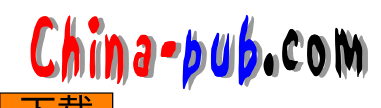
`[L,U]=lu(A)`	求上三角矩阵 **U**和交换下三角矩阵 **L**。**L**是一个带有单位对角线的下三角矩阵和交换矩阵，即**P**的逆矩阵的乘积，见下个命令。
`[L,U,P]=lu(A)`	求上三角矩阵 **U**、有单位对角线的下三角矩阵 **L**和交换矩阵 **P**，满足**LU=PA**。

■ 例7.4

如果矩阵**A** = [1 2 3 ; 4 5 6 ;7 8 0]，输入`[L,U]=lu(A)`，运行结果为：

```
L =
     0.1429     1.0000          0
     0.5714     0.5000     1.0000
     1.0000          0          0

U =
     7.0000     8.0000          0
          0     0.8571     3.0000
          0          0     4.5000
```



这里的交换矩阵的逆为：

$$\mathbf{P}^{-1}=\mathbf{P}'=\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix}$$

下面列出高斯消元法的详细步骤：

$$\mathbf{A}=\begin{pmatrix}1&2&3\\4&5&6\\7&8&0\end{pmatrix}\quad \mathbf{A1}=\begin{pmatrix}7&8&0\\4&5&6\\1&2&3\end{pmatrix}\quad \mathbf{A2}=\begin{pmatrix}7&8&0\\0&0.4286&6\\0&0.8571&3\end{pmatrix}$$

$$\mathbf{A3}=\begin{pmatrix}7&8&0\\0&0.8571&3\\0&0.4286&6\end{pmatrix}\quad \mathbf{A4}=\begin{pmatrix}7&8&0\\0&0.8571&3\\0&0&4.5\end{pmatrix}$$

先从矩阵**A**开始，第1个主元的位置在(1, 1)，通过第1行和第3行互换，可以得到最大的主元，也就是7。第1次交换后得到矩阵**A1**，第1次消元后得到矩阵**A2**。

第2个主元在(2, 2)上，将第2行和第3行交换后得到矩阵**A3**，消元后得到矩阵**A4**，和矩阵**U**是同一个矩阵。

这个过程进行了两次交换，第1次是第1行和第3行互换，也就是：

$$\mathbf{P1}=\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}$$

第2次进行第2、3行互换，也就是：

$$\mathbf{P2}=\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix}$$

P1和**P2**相乘，形成P：

$$\mathbf{P1}*\mathbf{P2}=\mathbf{P}=\begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix}$$

P的逆为：

$$\mathbf{P}^{-1}=\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix}$$

如果输入`[L,U,P]=lu(A)`，运行就会得到如下的结果：

```
L =
      1.0000          0          0
      0.1429     1.0000          0
      0.5714     0.5000     1.0000

U =
      7.0000     8.0000          0
           0     0.8571     3.0000
           0          0     4.5000
```




```
P =
     0     0     1
     1     0     0
     0     1     0
```

■

有许多方法可求解 **Ax=b**这样的问题，**A**是$n \times n$的矩阵，**b**是一个长度为n的列向量。在本书中没有对算法进行完整的描述，详细信息可参见 MATLAB Help Desk。

命令集70　解方程组的方法

`x=bicg(A,b,`*`tol`*`,`*`maxit`*`,`*`M`*`)`	用双共轭梯度法解方程组。如果给定 *tol*，则用它来指定解的精度，也就是和`norm(b−A*x)/norm(b)`比较来判断解是否可以接受。如果给定 *maxit*，则用它作为最大的迭代数。要使用预处理因子，可在矩阵**M**中规定它。
`bicg(A,b,···,M1,M2,`*`x0`*`)`	操作同上，但是使用矩阵**M1**和**M2**作为预处理因子。矩阵**M1**、**M2**和预处理因子**M**的关系为**M=M1·M2**。如果给定**x0**，则将它作为初始化向量开始迭代。
`[x,flag,relres,iter,resvect]=bicg(···)`	求**x**中的问题解，有关*bigc*的收敛信息存放在*flag*中。结果的相对剩余范数保存在*iter*中。值*resvect*是每次迭代的范数，结果向量中的所有变量都可忽略。
`bicgstab(···)`	用稳定双共轭梯度法解方程组，调用方式和返回的结果形式和命令`bicg`一样。
`cgs(···)`	用复共轭梯度平方法解方程组，调用方式和返回的结果形式和命令`bicg`一样。
`gmres(A, b, restart, ···)`	用广义最小残量法解方程组，除了参数 *restart*可以给出外，调用方式和返回的结果形式和命令`bicg`一样。
`pcg(···)`	用预处理共轭梯度法解方程组，调用方式和返回的结果形式和命令`bicg`一样。
`qmr(···)`	用准最小残量法解方程组，调用方式和返回的结果形式和命令`bicg`一样。

7.3　行梯形矩阵

LU因式分解的另一种方法就是将系数矩阵 **A**降维成行梯形矩阵的形式。因为它能应用在长方矩阵上，所以这种方法更常用。这种矩阵能给出许多有关线性系统的信息。

对于每一个$m \times n$的矩阵**A**都有一个交换矩阵**P**，一个有单位对角线的下三角矩阵 **L**和一个$m \times n$的梯形矩阵**R**，它们满足**PA=LR**。

如果下列情况成立，则矩阵是梯形矩阵：

1) 如果有零行，就放在矩阵的底部；

2) 每行中第一非零元素是1，它作为每行的最主要元素；

3) 每行的最主要元素放在上一行最主要元素的右边；
4) 在有最主要元素1的列中，其他元素都是零。
可以用MATLAB中命令rref来分析系统**Ax=b**。

命令集71 缩减行阶梯矩阵

`rref(A)`	用高斯—约当消元法和行主元法求 **A**的缩减行的阶梯矩阵。
`rref(A,tol)`	和rref(A)一样，但是使用精度 *tol*，它用来决定什么时候元素可以忽略不计。
`rrefmovie(A)`	求缩减行的阶梯矩阵，并给出每一步的求解过程。

通常使用的精度是`tol=max(size(A))*eps*norm(A, inf)`，在7.6节中定义了命令norm。

■ **例**7.5

令

$$\mathbf{A}=\begin{pmatrix}1 & 3 & 3 & 2\\ 2 & 6 & 9 & 5\\ -1 & -3 & 3 & 0\end{pmatrix}\quad \mathbf{B}=\begin{pmatrix}2 & 1 & 1\\ 4 & 1 & 0\\ -2 & 2 & 1\end{pmatrix}$$

(a) 如果用命令rref(A)得到的矩阵中有一个或多个零行，则可以知道矩阵 **A**中有一些行线性相关。同样也可用命令`rref(A´)` 来知道矩阵**A**的列向量的相关性。

输入命令`Aref=rref(A)`，`Bref=rref(B)`，运行结果为：

```
Aref =
    1.0000    3.0000         0    1.0000
         0         0    1.0000    0.3333
         0         0         0         0

Bref =
     1     0     0
     0     1     0
     0     0     1
```

(b) 求矩阵**A**的秩可以在**A**的缩减行的阶梯矩阵中数非零行的数量。在这个例子中，可以知道**A**的秩是2，**B**的秩是3。可以用命令`rankA=rank(A)`，`rankB=rank(B)`来确认一下：

```
rankA =
         2

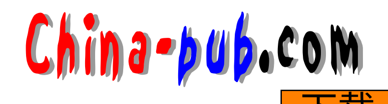
rankB =
         3
```

■

命令`rref`可以用来研究线性方程组，令 **Ax=b**代表一个线性方程组，令矩阵 **B**=(A b)。用命令`C=rref(B)`求得矩阵**B**的缩减行的阶梯矩阵**C**，然后有下列结论：

• 如果**C**中有一个或多个全零的行向量，则这个方程组中有冗余信息。这就意味着可以去掉一个或多个方程

• 如果**C**中有除了最后一个元素不是零外其余都是零的行，即(0, 0, ···,0, 1)，则这个方程组无解



• 如果方程组有唯一解，则在C的最后一个列可以得到该解。

7.4 Cholesky因式分解

如果矩阵**A**是一个对称正定矩阵，即**A**=**A**′且对于每个**x**≠0都有x′**Ax**>0，则存在一个上三角矩阵**G**，它的对角线元素是正数，且满足**G**′**G**=**A**。

这是LU因式分解的一种特殊情况称为Cholesky因式分解，它只有标准LU因式分解的大约一半的计算步骤。注意，在有些书中是按照下三角矩阵来定义Cholesky因式分解的。

在用左除\来解对称正定的方程组时，MATLAB会自动用Cholesky因式分解来求解。

命令chol可用来计算正定矩阵**A**的Cholesky因式分解。

命令集72 Cholesky因式分解

`chol(A)`	求矩阵**A**的Cholesky因子，是一个上三角矩阵。如果**A**不是一个正定矩阵，则给出一个错误信息。
`[G,err]=chol(A)`	求矩阵**A**的Cholesky因子**G**。如果**A**不是一个正定矩阵，则不给出错误信息，而是将err设为非零值。
`R1=cholupdate(R,x)`	如果**R**=chol(A)且x是一个和**A**的列长度一样的列向量，则求**A**+**xx**′的上三角Cholesky因子**R1**。
`R1=cholupdate(R,x,'-')`	如果R=chol(A)且x是一个和**A**的列长度一样的列向量，则求A－**xx**′的上三角Cholesky因子**R1**。

■ **例**7.6

```
(a) b = [-1 -1 -1]; A = 4*eye(4) + diag(b,-1) + diag(b,1),...
    G = chol(A)
```

结果为：

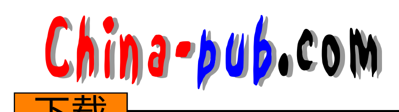
```
A =
     4    -1     0     0
    -1     4    -1     0
     0    -1     4    -1
     0     0    -1     4

G =
    2.0000   -0.5000         0         0
         0    1.9365   -0.5164         0
         0         0    1.9322   -0.5175
         0         0         0    1.9319
```

用Test=G′*G来检验这个结果，得：

```
Test =
    4.0000   -1.0000         0         0
   -1.0000    4.0000   -1.0000         0
         0   -1.0000    4.0000   -1.0000
         0         0   -1.0000    4.0000
```

这样就又得到矩阵**A**。



(b) 为了看清楚LU因式分解和Cholesky因式分解在计算步骤的不同，输入：

```
flops(0), lu(A); flops, flops(0), chol(A); flops
```

结果显示为：

```
ans =
        34
ans =
        30
```

■

对于较大的方程组来说，它们之间的差别更加明显。

7.5 QR因式分解

解线性方程组除了LU因式分解和Cholesky因式分解，还有第三种方法即QR因式分解或QR分解。

假设**A**是$n \times n$的矩阵，那么**A**就可以分解成：

```
A=QR
```

其中**Q**是一个正交矩阵，**R**是一个大小和**A**相同的上三角矩阵，因此**Ax=b**可以表示为**QRx=b**或者等同于：

```
Rx=Qb
```

这个方程组的系数矩阵是上三角的，因此容易求解。

和高斯消元法比较，QR因式分解的主要优点在于有更高的稳定性，然而它的数学运算更麻烦一些。

MATLAB中用命令qr来求QR因式分解，这个命令可以分解$m \times n$的矩阵，所以考虑一般情况，假设**A**是$m \times n$的矩阵。

命令集73　QR因式分解

`[Q,R]=qr(A)`	求得$m \times m$的矩阵**Q**和上三角矩阵**R**，**Q**的列形成了一个正交基，**Q**和**R**满足**A=QR**。
`[Q,R,P]=qr(A)`	求得矩阵**Q**、上三角矩阵**R**和交换矩阵**P**。**Q**的列形成一个正交基，**R**的对角线元素按大小降序排列，它们满足**AP=QR**。
`[Q,R]=qr(A,0)`	求矩阵**A**的QR因式分解。如果在$m \times n$的矩阵**A**中行数小于列数，则给出**Q**的前n列，因此**Q**的大小和**A**相同。也能得到交换矩阵，见上或输入help qr可获得帮助。
`[Q1,R1]=qrdelete(Q,R,j)`	求去掉矩阵**A**中第j列之后形成的矩阵的**QR**因式分解，矩阵**Q**和**R**是**A**的**QR**因子。
`[Q1,R1]=qrinsert(Q,R,b,j)`	求在矩阵**A**的第j列前插入一列向量**b**后形成的矩阵的**QR**因式分解，矩阵**Q**和**R**是**A**的**QR**因子。如果$j=n+1$，那么插入的一列放在最后。

■ **例7.7**

(a) 解**Ax=b**，其中**A**和**b**给出如下：

$$\mathbf{A}=\begin{pmatrix}1 & 2 & 2\\ 3 & 2 & 2\\ 1 & 1 & 2\end{pmatrix}\qquad \mathbf{b}=\begin{pmatrix}7\\ 9\\ 5\end{pmatrix}$$

```
[Q,R] = qr(A), x = R\Q'*b
```

结果为：

```
Q =
    -0.3015    0.9239   -0.2357
    -0.9045   -0.3553   -0.2357
    -0.3015    0.1421    0.9428

R =
    -3.3166   -2.7136   -3.0151
          0    1.2792    1.4213
          0         0    0.9428

x =
     1
     2
     1
```

这个结果和用命令A\b得到的结果一样。

(b) 下面比较一下用 QR因式分解法和左除法解线性方程组的计算步骤。为了比较明显，用它们来解一个较大的方程组：

```
A = rand(10,10); b = rand(10,1); flops(0); ...
x = A\b; lureq = flops
```

结果为：

```
lureq =
        1506
```

和

```
flops(0), [Q,R] = qr(A); x = R\Q'*b; qrreq = flops
```

结果为：

```
qrreq =
        5285
```

■

QR因式分解能用来解超定方程组，这样的方程组中方程的个数比未知变量的个数多 (见7.7节)，还能用来求特征值和特征向量。

计算矩阵QR因式分解的一种方法可以应用在 **Givens旋转**上。用命令planerot来求长度为2的向量的Givens平面旋转。必须用冒号表达式才能将 Givens平面旋转应用在矩阵上。

命令集74 Givens和Jacobi旋转

planerot(x)	求去掉有两个元素的向量 **x**中第1个元素的2×2矩阵的Givens旋转。在MATLAB函数qrinsert和qrdelete中使用这个命令。
[G,y]=planerot(x)	在**G**中返回Givens旋转，结果为**y**=**Gx**。
rjr(A)	求**A**的Jacobi旋转，旋转的角度和平面都是随机数。保存有特征向量、奇异值和对称性。



■ **例**7.8

Givens旋转可以描述成平面旋转，假设2×2的矩阵**A**有列向量**x**和**y**。

$$\mathbf{A}=\begin{pmatrix}1 & 1\\ -1 & 2\end{pmatrix}\qquad \mathbf{x}=\begin{pmatrix}1\\ -1\end{pmatrix}\qquad \mathbf{y}=\begin{pmatrix}1\\ 2\end{pmatrix}$$

如果输入`G=planerot(x)`，则**G**旋转向量**x**(**A**的第1列)成x轴，并用`Anew=G*A`对**A**操作，得到的结果为：

```
G =
     0.7071    -0.7071
     0.7071     0.7071

Anew =
     1.4142    -0.7071
          0     2.1213
```

用下列命令求出**A**和**Anew**的列向量范数是一样的：

```
xnorm = norm(x), ynorm = norm(y), ...
xnewnorm = norm(Anew(:,1)), ynewnorm = norm(Anew(:,2))


xnorm =
     1.4142

ynorm =
     2.2361

xnewnorm =
     1.4142

ynewnorm =
     2.2361
```

每列的范数(也称长度)是不变的，在7.6节定义了命令`norm`，从图7-1可以看出对于A中的两列旋转是一样的。

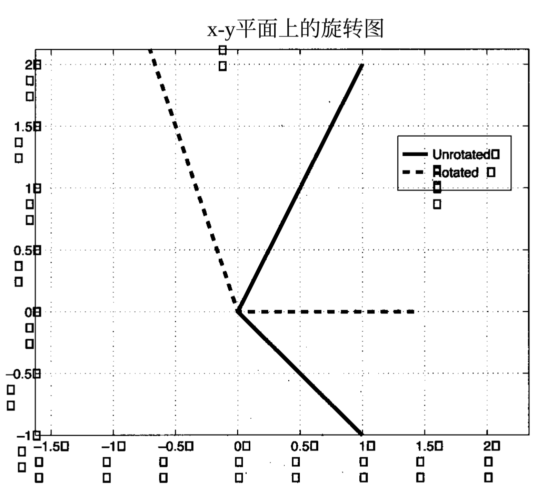


图7-1　在x-y平面上的两个向量的Givens旋转图

■



要编写一些程序才能将Givens旋转应用在矩阵上和进行缩减矩阵元素，可以参见12.2节中关于命令planerot的例题。

7.6 范数和条件数

向量的范数是一个标量，用来衡量向量的长度，不要和向量中元素的个数相混淆。MATLAB中可以用命令norm得到不同的范数。

命令集75 向量范数

norm(x)	求欧几里得范数，即 $\|\mathbf{x}\|_2=\sqrt{\sum\|x_k\|^2}$。
norm(x,inf)	求∞-范数，即||x||=max(abs(x))。
norm(x,1)	求1-范数，即 $\|\mathbf{x}\|_1=\sum_k\|x_k\|$。
norm(x,p)	求p-范数，即 $\|\mathbf{x}\|_p=\sqrt[p]{\sum_k\|x_k\|^p}$，所以norm(x, 2)=norm(x)
norm(x,−inf)	求向量**x**的元素的绝对值的最小值，即min(abs(x))。注意，这不是向量的范数。

■ **例7.9**

令

x=[3 4 5]

```
norm1   = norm(x,1),   norm2  = norm(x,2), ...
norminf = norm(x,inf), nonorm = norm(x,-inf)
```

结果为：

```
norm1 =
    12

norm2 =
    7.0711

norminf =
     5

nonorm =
     3
```

■

矩阵范数用来衡量矩阵大小，和矩阵的行、列数的概念是不一样的。由于数据或数字计算中的扰动，矩阵范数常用来估计误差。

用向量的范数来定义方阵的p-范数：

$$\|\mathbf{A}\|_p=\sup_{\mathbf{x}\neq\mathbf{0}}\frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_p}$$

定义的这些范数并不真正用在计算中，而是用命令集 76中的表达式，它们可以求方阵范数，也可以用来求非方阵的范数。

由于欧几里得范数计算很复杂，所以用命令normest来计算欧几里得范数的估计值。

命令集76　矩阵范数

`norm(A)`	求欧几里得范数$\|\|A\|\|_2$，等于**A**的最大奇异值，参见8.3节。
`norm(A,1)`	求列范数$\|\|A\|\|_1$，等于**A**的列向量的1-范数的最大值。
`norm(A,2)`	求欧几里得范数$\|\|A\|\|_2$，和`norm(A)`一样。
`norm(A,inf)`	求行范数$\|\|A\|\|_\infty$，等于**A**的行向量的1-范数的最大值。
`norm(A,'fro')`	求Frobenius范数 $\|\mathbf{A}\|_F = \sqrt{\sum_i \sum_j \|a_{ij}\|^2}$，这不能用矩阵$p$-范数的定义来求。
`normest(A)`	求欧几里得范数的估计值，相对误差小于10^{6}。
`normest(A,tol)`	求欧几里得范数的估计值，相对误差小于*tol*。

■ **例**7.10

(a) 令

```
A = [1 1;2 3];

norm1   = norm(A,1),   norm2 = norm(A,2), ...
norminf = norm(A,inf), normf = norm(A,'fro')
```

结果为：

```
norm1 =
     4

norm2 =
    3.8643

norminf =
     5

normf =
    3.8730
```

(b) 求一个大矩阵的欧几里得范数和欧几里得范数估计值，并对它们进行比较：

```
A = rand(100);
flops(0), norm2 = norm(A), expensive = flops
```

结果为：

```
norm2 =
   49.8483

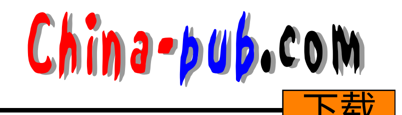
expensive =
     2948409
```

和

```
flops(0), normapprox = normest(A), cheaper = flops
```

结果为：

```
normapprox =
```




```
    50.6701

cheaper =
       133211
```

■

令**Ax=b**表示线性方程组。方程组的条件数是一个大于或者等于 1的实数，用来衡量关于数据中的扰动，也就是 **A**和/或**b**，对解**x**的灵敏度。一个差条件的方程组的条件数很大。条件数的定义为：

$$\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$$

命令cond(A)可求出欧几里得范数的条件数，就是**A**的最大奇异数和最小奇异数的商；参见8.3节。

命令集77 条件数

`cond(A)`	求**A**的欧几里得范数的条件数。
`cond(A,p)`	求*p*-范数的条件数，*p*的值可以是1、2、*inf*或者'*fro*'；见命令集76中命令norm。
`condest(A)`	求矩阵**A**条件数的1-范数中的下界估计值。
`[c,v]= condest(A)`	求**A**的1-范数中条件数c的下界估计值，同时还计算向量v，使得它们满足条件 $\|\mathbf{A}v\| = \frac{\|A\|\|v\|}{c}$。
`[c,v]= condest(A,tr)`	求如上的*c*和**v**，同时显示出关于计算的步骤信息。如果*tr*=1，则计算的每步都显示出来；如果*tr*=-1，则给出商c/rcond(A)。
`rcond(A)`	求矩阵**A**定义的方程组的敏感度的另一个估计值。对于差条件矩阵**A**来说，给出一个接近于 0的数；对于好条件矩阵**A**，则给出一个接近于1的数。

■ **例**7.11

求Hilbert矩阵的条件数(见4.4节)：

```
bad=cond(hilb(5))
```

结果为：

```
bad =
    4.7661e+05
```

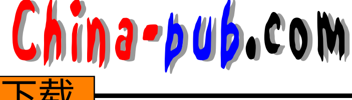
这表明在最坏情况下右边或者系数矩阵的一个扰动和数bad相乘以后，可能会丢失五位小数。

■

7.7 超定方程组和欠定方程组

对于一个系数矩阵是*m*×*n*的线性方程组**Ax=b**来说，如果*m*>*n*，也就是说方程的个数多于未知数，则称为超定方程组。通常这些方程组是矛盾的，所以方程组没有精确解。在拟合实验数据的曲线时，常会遇到这个问题。

关键是要找到一个向量 **x**使它对*m*个方程的总误差最小。有几种方法可以求得，但是最常用的方法是最小二乘法：



$$e=\sum_{i=1}^{m}\left(b_i-\sum_{j=1}^{n}a_{ij}x_j\right)^2=\|\mathbf{b}-\mathbf{Ax}\|_2^2$$

这就是最小二乘法，最小二乘解可以用除法运算符 \或命令nnls和lscov来解得。这种方法适用于在第9章中讨论的稀疏矩阵，下面提到了在计算中创建稀疏矩阵的命令 spaugment。

命令集78　最小二乘解

`A\b`	求最小二乘解，也可参见 3.3节。如果**b**=**B**是一个矩阵，则是对和**B**中每一列相对应的方程求解。
`spaugment(A,c)`	生成一个对称的稀疏方阵T=[c*I A; A´0]，可以用T\z来解超定方程组**Ax=b**，其中**z**是带有尾随零的向量**b**。参数c可以省去，MATLAB就根据spparms中的设定来使用值。输入help spparms 可得更多帮助。
`nnls(A,b)`	求非负最小二乘解，输入help nnls可得到更多帮助。
`lscov(A,b,v)`	求在已知协方差V情况下的最小二乘解，这意味着(**b**－**Ax**′)′ ($\mathbf{V}^{-1}$(**b**－**Ax**)最小，输入help lscov 可得到更多帮助。

MATLAB在用左除解超定方程组时都用 QR因式分解。如果方程组的系数矩阵不满秩，就没有唯一的最小二乘解，将不定值设为零。然后给出一个警告信息。

■ **例**7.12

令

$$\mathbf{A1}=\begin{pmatrix}1&1\\1&2\\1&3\end{pmatrix}\qquad\mathbf{A2}=\begin{pmatrix}1&3\\1&3\\1&3\end{pmatrix}\qquad\mathbf{b}=\begin{pmatrix}2\\0\\-1\end{pmatrix}$$

```
fullrank = A1\b, notfullrank = A2\b
```

结果为：

```
fullrank =
    3.3333
   -1.5000

Warning: Rank deficient, rank = 1  tol  = 3.4613e-15.

notfullrank =
         0
    0.1111
```

■

如果$n>m$，则方程组**Ax=b**是欠定方程组。这样的方程组通常有无穷组解，MATLAB在求解时给出一组解，不给出警告信息。

■ **例**7.13

(a) 令

$$\mathbf{A}=\begin{pmatrix}1&1&1\\1&2&3\end{pmatrix}\qquad\mathbf{b}=\begin{pmatrix}2\\3\end{pmatrix}$$

```
x = A\b
```

解为：

```
x =
    1.5000
         0
    0.5000
```

不幸的是MATLAB不能给出它的通解，用命令rref来研究方程组的可解性，参见7.3节。输入：

```
UnderSol = rref([A~b])
```

得到结果为：

```
UnderSol =
     1     0    -1     1
     0     1     2     1
```

从中可以找出它的通解，如：

$$\mathbf{x} = \begin{pmatrix} 1+t \\ 1-2t \\ t \end{pmatrix} \qquad \text{对所有的} t$$

t=0.5时就是MATLAB求得的解。

(b) 令

$$\mathbf{a} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 2 \\ -2 \\ 2 \end{pmatrix}$$

下面比较一下矩阵的各种除法所得结果， **a**除以**b**。

左除的结果是：

```
left = a\b

left =
   0.2857
```

或者2/7，这就是超定方程组**ax=b**的解。

右除的结果是：

```
Right = b/a

Right =
         0         0    0.6667
         0         0   -0.6667
         0         0    0.6667
```

这和(a ' \b ')结果一样，这是欠定方程组**a** ' **x** ' =**b** ' 的一个解。

数组的左除：

```
elementWiseLeft = a.\b

elementWiseLeft =
    2.0000
   -1.0000
    0.6667
```

结果和右除是一样的：

```
elementWiseRight = b./a

elementWiseRight =
    2.0000
   -1.0000
    0.6667
```



第8章　特征值和特征向量

MATLAB中的命令计算特征值和特征向量很方便，可以得到不同的子结果和分解，这在线性代数教学时很有用。注意，本章中的命令只能对二维矩阵操作。

8.1　特征值和特征向量的计算

假设**A**是一个$m \times n$的矩阵，**A**的特征值问题就是找到方程组的解：

$$\mathbf{Ax} = \lambda \mathbf{x}$$

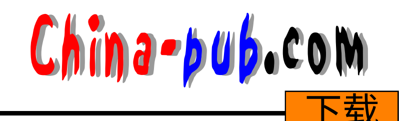
其中λ是一个标量，**x**是一个长度为n的列向量。标量λ是**A**的特征值，**x**是相对应的特征向量。对于实数矩阵**A**来说，特征值和特征向量可能是复数。一个$n \times n$的矩阵有n个特征值，表示为λ_1，λ_2，…，λ_n。

MATLAB中用命令eig来确定矩阵**A**的特征值和特征向量。特征向量的规格化，就是每个特征向量的欧几里得范数为1；参见7.6节。

命令eig自动完成对矩阵**A**的平衡化。这就要求MATLAB找出一个**相似变换矩阵Q**，满足条件$\tilde{\mathbf{A}}$。求$\tilde{\mathbf{A}} = \mathbf{Q}^{-1}\mathbf{AQ}$的特征值比求A的特征值条件更好些。万一**A**有一个和机器错误大小一样的元素，平衡化对于计算过程是没有好处的。带有参数 `nobalance`的命令`eig`可用来计算没有这个变换矩阵的特征值和特征向量。

命令集79　特征值和特征向量

`eig(A)`	求包含矩阵**A**的特征值的向量。
`[X,D]=eig(A)`	产生一个矩阵**A**的特征值在对角线上的对角矩阵**D**和矩阵**X**，它们的列是相应的特征向量，满足**AX**=**XD**。为了得到有更好条件特征值的矩阵要进行相似变换。
`[X,D]=eig(A,'nobalance')`	不经过平衡处理求得矩阵**A**的特征值和特征向量，也就是不进行平衡相似变换。
`balance(A)`	求平衡矩阵。
`[T,B]=balance(A)`	找到一个相似变换矩阵**T**和矩阵**B**，使得它们满足$\mathbf{B}=\mathbf{T}^{-1}\mathbf{AT}$。**B**是用命令`balance`求得的平衡矩阵。
`eigs(A)`	返回一个由矩阵**A**的部分特征值组成的向量，和命令`eig`一样，但是不返回全部的特征值。如果不带有参量，则计算出最大的特征值。当计算所有特征值时，如果矩阵**A**的秩不小于6，则计算出6个特征值来。
`eigs(f,n)`	求出矩阵**A**的部分特征值。在使用一个矩阵列的线性运算符时，字符串**f**中包含的是**M**文件的文件名，n指定问题的阶次。用这种方法来求特征值比开始就用运算符来求要快。



eigs(A,*B*,*k*,*sigma*)	求矩阵**A**的部分特征值，矩阵**B**的大小和**A**相同；如果没有给出**B**=eye(size(A))，那么*k*就是要计算的特征值的个数；如果*k*没有给出，就用小于6的数或者**A**的秩；变量*sigma*是一个实数或者复数的移位参数，或者下列文本字符串中的一个，文本字符串指明需要的是哪种特征值： ‘lm’最大的特征值(缺省) ‘sm’最小的特征值 ‘lr’最大的实数部分 ‘sr’最小的实数部分 ‘be’同时求得最大和最小的实数部分
condeig(A)	返回一个由矩阵**A**的特征值条件数组成的向量。
[V,D,s]=condeig(A)	返回[V,D]=eig(A)和s=condeig(A)。

如果A是实数矩阵，MATLAB在计算中用QR因式分解；否则用QZ因式分解。

左特征向量是满足下面条件的非零行向量 y：

yA=λy

如果用命令eig对**A**′作用，也可以计算出左特征向量，因为：

$$\mathbf{A}'\mathbf{y}' = \bar{\lambda}\mathbf{y}'$$

这里的撇号'代表矩阵的转置和共轭复数(见3.4节)，λ上的短杠表示共轭复数。矩阵特征值的集合称为矩阵的谱，谱半径 ρ(A)定义为max(abs(eig(A)))。矩阵**A**的特征值的乘积等于det(A)，和等于trace(A)，这是矩阵**A**主对角线上元素的和。

如果**X**一个列向量为**A**的特征向量的矩阵，并且它的秩为 n，那么特征向量线性无关。如果不是这样，则称矩阵为缺陷阵。如果 **X'X**=**I**，则特征向量正交，这对于对称矩阵是成立的。

■ **例**8.1

矩阵**A**定义为：

$$\mathbf{A} = \begin{pmatrix} -9 & -3 & -16 \\ 13 & 7 & 16 \\ 3 & 3 & 10 \end{pmatrix}$$

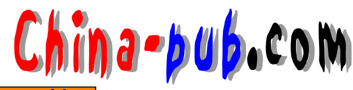
(a) 运行命令[Evect,Evalue]=eig(A)，得到结果为：

```
Evect =
   -0.7071   -0.5774   -0.5774
    0.7071    0.5774   -0.5774
    0.0000    0.5774    0.5774

Evalue =
   -6.0000         0         0
         0   10.0000         0
         0         0    4.0000
```

可知特征值都是非零数，矩阵是满秩的，可以用 therank=rank(Evect)来确认：

```
therank=
     3
```


下载

令M=Evect'*Evect得：

```
M =
     1.0000    0.8165   -0.0000
     0.8165    1.0000    0.3333
    -0.0000    0.3333    1.0000
```

可知特征向量没有相互正交。

(b)
```
determinant = prod(diag(Evalue)), ...
determinant2 = det(A)
```

给出结果为：

```
determinant =
 -240.0000

determinant2 =
  -240
```

可知行列式等于特征值的积。

(c)
```
theTrace = trace(A), theTrace2 = sum(diag(Evalue))
```

结果为：

```
theTrace =
     8

theTrace2 =
    8.0000
```

■

可知矩阵的迹等于特征值的和。

如果矩阵**A**是实数矩阵，但是有复数特征值，那么这些特征值是以共轭复数的形式出现的。如果[X,D]=eig(A)，可以用命令cdf2rdf将矩阵**D**转换为一个实数块对角矩阵。在对角线上用一个2×2实数块代替共轭复数对。

命令集80　**复对角矩阵变成实对角矩阵**

`[Y,E]=cdf2rdf(X,D)`	将复对角矩阵**D**变成实对角矩阵**E**，**Y**的列不是**A**的特征向量。

■ **例8.2**

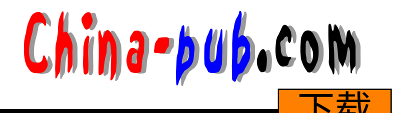
假设矩阵A为：

$$\mathbf{A}=\begin{pmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 3\end{pmatrix}$$

则运行[X,D]=eig(A)可得：

```
X =
   0.7071                 0.7071                   0
        0 + 0.7071i            0 - 0.7071i         0
        0                      0              1.0000

D =
        0 + 1.0000i            0                   0
        0                      0 - 1.0000i         0
        0                      0              3.0000
```



X和D都是复数矩阵，运行命令：[Y,E]=cdf2rdf(X,D)，结果为：

```
Y =
    0.7071         0         0
         0    0.7071         0
         0         0    1.0000

E =
     0     1     0
    -1     0     0
     0     0     3
```

所得的矩阵E正好是A矩阵。

■

注意 特征向量是特征多项式det(λI－A)=0的根，其中I是单位矩阵。用命令poly来求特征多项式，参见11.1节。命令roots也可求得特征值，但是用命令eig求得的特征值更准确，精度更高。

广义特征值问题就是找到方程组**Ax**=λ**Bx**的重要解，其中**B**也是一个$n \times n$的矩阵。λ值和向量**x**分别称为广义特征值和广义特征向量。

如果**B**是一个奇异矩阵，则用**QZ算法**来求解。

标准和广义特征值问题都属于矩阵多项式特征问题，都可以用命令polyeig来求它们的解。

命令集81 广义特征值和广义特征向量

eig(A,B)	返回一个含有广义特征值的向量，**A**和**B**都是方阵。
[X,D]=eig(A,B)	返回一个对角线上为广义特征值的对角矩阵 **D**和矩阵**X**，**X**的列是相对应的特征向量，因此有 **AX=BXD**。
[X,v]= polyeig(A0,A1,...,AK)	给出度为k的特征问题$(\mathbf{A}_0+\lambda\mathbf{A}_1+\lambda\mathbf{A}_2+...+\lambda^k\mathbf{A}_k)\mathbf{x}=\mathbf{0}$的特征值和特征向量。向量**v**的长度为**nk**，包含有特征值；$n \times nk$的矩阵**X**的列是特征向量。如果有 $\mathbf{A}_0$=**A**和$\mathbf{A}_1$=－I，那么这就是标准特征值问题。

为了检查特征值的条件或者它的敏感性，可以计算出条件数 $\text{cond}(\mathbf{X})=\|\mathbf{X}\|\,\|\mathbf{X}^{-1}\|$，矩阵**X**的列是**A**的特征向量。条件数大表示坏条件，也就是对扰动很敏感。

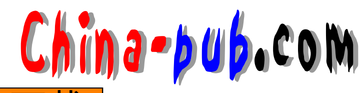
为了检查特征向量的条件或者它的敏感性可以查看特征值，多个重复的特征值或者特征值彼此相差很小就表示是坏条件问题。

■ **例**8.3

假设：

$$\mathbf{A}=\begin{pmatrix} 3.75 & -0.5 & -0.375 & 0.495 & -1.37 \\ 0.25 & 2.5 & 0.375 & -0.495 & -0.63 \\ 1.25 & -0.5 & 2.875 & 0.495 & -2.12 \\ 0.25 & -0.5 & -0.625 & 2.505 & 0.37 \\ 0.25 & -0.5 & -0.625 & 0.495 & 2.38 \end{pmatrix}$$

运行命令[XX, DD]=eig(A)结果为：




```
XX =
 -0.0000   0.0000 - 0.4472i   0.0000 + 0.4472i  -0.4472   0.4472
  0.5000  -0.0000 + 0.4472i  -0.0000 - 0.4472i  -0.4472  -0.4472
  0.5000   0.0000 - 0.4472i   0.0000 + 0.4472i  -0.4472   0.4472
 -0.5000   0.0000 - 0.4472i   0.0000 + 0.4472i  -0.4472  -0.4472
 -0.5000   0.0000 - 0.4472i   0.0000 + 0.4472i  -0.4472   0.4472
```

显然两个特征向量，列2和列3是复数。

```
DD =
  4.0000        0                   0                  0        0
      0    3.0000 + 0.0000i         0                  0        0
      0         0             3.0000 - 0.0000i         0        0
      0         0                   0             2.0000        0
      0         0                   0                  0   2.0100
```

从上可以看出特征值为2，2.01，3，3和4，可以知道这个特征值问题是一个坏条件问题。输入`badMatrix=cond(XX)`可求得条件数：

```
badMatrix =
   5.0156e+07
```

将这个数和例8.2中矩阵的特征值条件数`niceMatrix=cond(X)`比较：

```
niceMatrix =
             1.0000
```

它们是不同的。

■

8.2　上海森伯形式、QR和QZ因式分解

如果只求特征值和特征向量，推荐用上一节中提到的方法。然而，有时要求更详细了解计算过程，可用在这一节和下一节中定义的命令来满足这样的要求。

如果矩阵**H**的第一子对角线下元素都是零，则它是一个**上海森伯(Hessenberg)矩阵**。如果矩阵是对称矩阵，则它的海森伯形式是对角三角阵。MATLAB可以通过相似变换将矩阵变换成这种形式。

命令集82　上海森伯形式

`hess(A)`	返回矩阵**A**的上海森伯形式。
`[P,H]=hess(A)`	返回一个酉矩阵**P**和上海森伯矩阵**H**，使**A**=**P H P**和**PP**´=**I**。

在MATLAB中，QR算法是计算矩阵所有特征值的一种有效的数学方法，也可以用命令`eig`来求。在用这种方法时，建议将矩阵转换成相似的上海森伯形式，参见例 8.4。

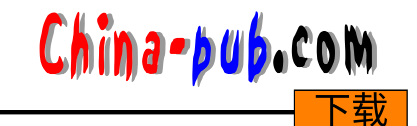
QR算法是基于QR因式分解的一种算法，每个$m\times n$的矩阵A可以表示成：

A=QR

其中**Q**是一个$m\times m$的酉矩阵，**R**是一个$m\times n$的上三角矩阵。如果**A**是一个方阵，**R**也还是这样的一个矩阵。当用命令`qr`时，会返回矩阵**Q**和**R**，也可参见例7.7。

命令集83　QR因式分解

`[Q,R]=qr(A)`	产生一个$m\times m$的酉矩阵**Q**和一个$m\times n$的上三角矩阵



	R，使得**A**=**QR**。
`[Q,R,P]=qr(A)`	产生一个大小为 $m\times m$、列正交的酉矩阵 **Q**，一个对角线元素递减的 $m\times n$ 的上三角矩阵 **R** 和一个置换矩阵 **P**，使得**AP**=**QR**。
`[Q,R]=qrinsert(Q,R,j,b)`	由于在矩阵**A**的*j*列后插入一个额外的列*b*而得到新的**QR**因式分解，**Q**和**R**是对矩阵A进行**QR**因式分解得到的矩阵。如果 $j=n+1$，那么*b*就插入在矩阵 **A**的最后一列。
`[Q,R]=qrdelete(Q,R,j)`	由于去掉矩阵**A**的第*j*列而得到新的**QR**分解，**Q**和**R**是对矩阵**A**进行**QR**分解得到的矩阵。
`[Q1,R1]=qrupdate(Q,R,x,y)`	给出**A**+**xy**´的**QR**分解，也就是用秩为1的矩阵改变**A**的**QR**分解。

如果**A**是上海森伯矩阵，则**Q**也是一样。对于**QR**算法，下面给出一些简短的描述：

QR**算法**：

1) 令 $\mathbf{A}_0=\mathbf{A}$，$k=0$；

2) 找到 $\mathbf{A}_k$ 的分解：$\mathbf{A}_k=\mathbf{Q}_k\mathbf{R}_k$；

3) 迭代计算下一个矩阵：$\mathbf{A}_{k+1}=\mathbf{Q}_k\mathbf{R}_k$，令 $k=k+1$；

4) 返回到2。

这种方法也称为不移位的QR方法，就是在某种约定下逼近于上三角矩阵。因为所有的矩阵 $\mathbf{A}_k$ 和 $\mathbf{A}_0=\mathbf{A}$ 相似，所以有和原始矩阵相同的特征值，即最后的上三角矩阵的对角线元素就是**A**的特征值。

如果矩阵一开始就转换成有接近一半元素是零的上海森伯形式，就可以减少可观的计算步骤。QR方法作为MATLAB的一个内建函数，为了加快逼近速度也可进行移位。

■ **例**8.4

用不移位的QR因式分解算法，计算例8.1中矩阵**A**的特征值

$$\mathbf{A}=\begin{pmatrix} -9 & -3 & -16 \\ 13 & 7 & 16 \\ 3 & 3 & 10 \end{pmatrix}$$

正确的特征值为 $\lambda_1=10$，$\lambda_2=4$ 和 $\lambda_3=-6$。

第1步：

```
A0 = hess(A); [Q0,R0] = qr(A0); A1 = R0*Q0
```

返回得到：

```
A1 =
     1.7992    26.8770   -12.6126
     2.3625     4.5085    -0.1434
          0     4.9518     1.6923
```

第2步：[Q1, R1]=qr(A1)；A2=R1*Q1，得到：

```
A2 =
   17.6077    11.3432     5.0128
  -15.3516   -13.6557    -5.8721
         0     1.0748     4.0480
```

在计算开始时，看不出要逼近什么样的矩阵。但是在计算了 10步以后，就可看出主对角线以下的元素较小。

```
[Q9,R9] = qr(A9); A10 = R9*Q9
```

结果为：

```
A10 =
   10.1297    22.6238    15.3505
   -0.0924    -6.1616    -5.8036
         0     0.0562     4.0319
```

■

注意，在整个计算过程中一直保留着上海森伯形式。

上例中的迭代过程可用MATLAB的内建编程语言简明地写出。在12.2节中有相关的例子。

QZ算法是用来计算复数矩阵的复数特征向量对和广义特征值的。在 MATLAB中，依据下面的命令集84来调用命令qz。

命令集84　**QZ算法**

`[C,D,Q,Z,V]=qz(A,B)`	得到对角线元素是广义特征值的上三角矩阵**C**、**D**和广义特征向量矩阵**V**。矩阵**Q**和**Z**是变换矩阵，使得**QAZ**=**C**和**QBZ**=**D**。

QZ方法是基于QZ分解的方法。

8.3　舒尔分解和奇异值分解

如果**A**是一个方阵，则有一个这样的酉矩阵 **U**，使得：

$$\mathbf{U}^{-1}\mathbf{AU} = \mathbf{U}'\mathbf{AU} = \mathbf{T}$$

其中T是上三角矩阵。这是一个相似变换，因此矩阵 **A**和**T**有相同的特征值。因为**T**是一个对角矩阵，因此其对角线元素就是它的特征值。

如果**A**是实数矩阵且对称，那么 **T**有对角形式， **U**的列就是**A**的特征向量。

如果**A**是实数矩阵，但是有复数特征值，那么 **T**就是一个复数矩阵。为了避免复杂的计算，用2×2的实数矩阵来代表每一对共轭复数特征值，参见例 8.2。这样**T**就是一个块三角实数矩阵。MATLAB中用命令`schur`来进行实数和复数矩阵**A**的舒尔分解。

如果**A**是实数矩阵，那么`schur(A)`返回实数的舒尔形式，但是如果 **A**是复数矩阵，则给出复数形式。它们的差别在于实数形式中在对角线上用 2×2的矩阵块来表示共轭复数特征值对，而复数形式给出的对角线元素是复数。函数 `rsf2csf`可以将实数形式转换成复数形式。

命令集85　**舒尔分解**

`schur(A)`	给出矩阵**A**的舒尔分解，也就是如上所述的矩阵 **T**。
`[U,T]=schur(A)`	给出矩阵**A**的舒尔分解和一个酉矩阵**U**，使得**A**=**UTU**′。
`[V,S]=rsf2csf(U,T)`	将实数舒尔形式矩阵**U**和**T**转变成复数舒尔形式矩阵**V**和**S**。

■ **例**8.5

将**A1**，**A2**，**A3**定义为：

$$\mathbf{A1}=\begin{pmatrix}2&1\\1&2\end{pmatrix}\quad \mathbf{A2}=\begin{pmatrix}2&1\\0&2\end{pmatrix}\quad \mathbf{A3}=\begin{pmatrix}0&1\\-1&0\end{pmatrix}$$

运行的命令为：

```
Sch1 = schur(A1), Sch2 = schur(A2)}, ...
[U,Sch3] = schur(A3)
```

结果为：

```
Sch1 =
      1      0
      0      3

Sch2 =
      2      1
      0      2
U =
      1      0
      0      1

Sch3 =
      0      1
     -1      0
```

因为有复数特征值，所以可以看出矩阵 **Sch3**不是一个上三角矩阵。为了验证一下可以输入：`[V,S]=rsf2csf(U,Sch3)`，得到：

```
V =
          0 + 0.7071i    0.7071
   -0.7071                    0 - 0.7071i

S =
          0 + 1.0000i         0
          0                   0 - 1.0000i
```

这里的特征值是i和$-i$。

■

MATLAB还可以计算奇异值分解，即 **SVD**和矩阵的奇异值。这些数都是非负数，在某种特定情况下，它们和矩阵的特征值相同。命令 svds和命令eigs相似，也返回一些奇异值。

命令集86　SVD**分解**

svd(A)	返回一个包含矩阵**A**奇异值的向量。
[U,S,V]=svd(A)	返回一个对角矩阵 **S**和大小分别为$m\times m$和$n\times n$的酉矩阵**U**和**V**，矩阵**S**的大小和 **A**相同，也是$m\times n$的，而且奇异值在矩阵的对角线上。奇异值是非负数且按降序

	排列。这些矩阵满足**A**=**USV**′和**U**′**AV**=**S**。
`[U,S,V]=svd(A,0)`	得到一个“有效大小”的分解，只计算出矩阵 **U**的前n列。矩阵**S**的大小为$n \times n$。
`svds(A,k,0)`	计算出k个最大的奇异值和相应矩阵 **A**的向量。如果k没有给出，则缺省值为 5。如果0作为最后一个参量值，则计算最小值；否则计算最大值。
`gsvd(A)`	给出广义奇异分解，参见gsvd的帮助可得更多相关信息。

在MATLAB中，矩阵**A**的伪逆可以用命令pinv(A)来求，也可用SVD分解来求矩阵的伪逆，参见7.1节。

如果s_i是奇异值，那么 $\|\mathbf{A}\|_2 = \max s_i = s_1$, $\|\mathbf{A}^{-1}\|_2 = (\min s_i)^{-1} = s_n^{-1}$ 和cond(A)=s_1/s_n，其中s_n是最小奇异值。对于非奇异矩阵和满秩的长方形矩阵($m>n$)，最后一个表达式都成立。

■ **例**8.6

假设矩阵**A**，**B**：

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} 1 & -1 & 0 \\ 1 & 2 & 0 \\ 1 & 3 & 0 \end{pmatrix}$$

(a) 求奇异值分解：

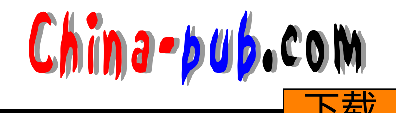
```
[Ua,Sa,Va] = svd(A), [Ub,Sb,Vb] = svd(B)

Ua =
     0.3231    -0.8538     0.4082
     0.5475    -0.1832    -0.8165
     0.7719     0.4873     0.4082

Sa =
     4.0791          0
          0     0.6005
          0          0

Va =
     0.4027    -0.9153
     0.9153     0.4027

Ub =
    -0.1641     0.9668     0.1961
     0.5653     0.2551    -0.7845
     0.8084     0.0178     0.5883
Sb =
     3.9116          0          0
          0     1.3036          0
          0          0          0
```




```
Vb =
    0.3092    0.9510         0
    0.9510   -0.3092         0
         0         0    1.0000
```

可以看出矩阵**A**和**B**有两个非零的奇异值，也就是它们的秩为2。

(b) 这些矩阵的逆是没有办法求的，但是伪逆可以用PseudoA=pinv(A)和PseudoB=pinv(B)来求：

```
PseudoA =
    1.3333    0.3333   -0.6667
   -0.5000    0.0000    0.5000

PseudoB =
    0.6923    0.2308    0.0769
   -0.2692    0.0769    0.1923
         0         0         0
```

(c) 矩阵**A**和**B**的范数和条件数为：

```
normA = norm(A), normB = norm(B), ...
condA = cond(A), condB = cond(B)

normA =
    4.0791

normB =
    3.9116

condA =
    6.7930

condB =
   Inf
```

可以看出欧几里得范数等于最大奇异值，条件数等于 s_1/s_n。

■



第9章 稀 疏 矩 阵

在许多问题中提到了含有大量0元素的矩阵，这样的矩阵称为稀疏矩阵。比如求解普通或者部分微分方程的数值解。为了节省存储空间和计算时间， MATLAB考虑到矩阵的稀疏性，在对它运算时有特殊的命令。

9.1 矩阵为什么稀疏

一个稀疏矩阵中有许多元素等于零，这便于矩阵的计算和保存。如果 MATLAB把一个矩阵当作稀疏矩阵，那么只需在 $m\times3$的矩阵中存储 m个非零项。第1列是行下标，第2列是列下标，第3列是非零元素值，不必保存零元素。如果存储一个浮点数要 8个字节，存储每个下标要4个字节，那么整个矩阵在内存中存储需要 $16\times m$个字节。

■ **例**9.1

```
A=eye(1000);
```

得到一个1000×1000的单位矩阵，存储它需要8 Mb空间。如果使用命令：

```
B=speye(1000);
```

用一个1000×3的矩阵来代表，每行包含有一个行下标、列下标和元素本身。现在只需16Kb的空间就可以存储1000×1000的单位矩阵，它只需要满单位矩阵的 0.2%存储空间。对于许多的广义矩阵也可这样来作。

■

稀疏矩阵的计算速度更快，因为 MATLAB只对非零元素进行操作，这是稀疏矩阵的第二个突出的优点。

■ **例**9.2

假设矩阵**A**、**B**和例9.1中的矩阵一样。计算 2*A需要一百万次的浮点运算，而计算 2∗B只需要2000次浮点运算。

■

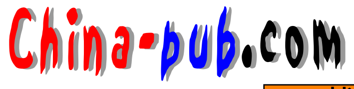
因为MATLAB不能自动创建稀疏矩阵，所以要用特殊的命令来得到稀疏矩阵，在下一节中将给出这些命令。前面章节中的算术和逻辑运算都适用于稀疏矩阵。

9.2 创建和转换稀疏矩阵

在MATLAB中，用命令`sparse`来创建一个稀疏矩阵。

命令集87 **创建稀疏矩阵**

`sparse(A)`	由非零元素和下标建立稀疏矩阵 **A**。如果**A**已是一个稀疏矩阵，则返回**A**本身。
`sparse(m,n)`	生成一个$m\times n$的所有元素都是0的稀疏矩阵。



`sparse(u,v,a)`	生成一个由长度相同的向量 **u**，**v**和**a**定义的稀疏矩阵。其中**u**和**v**是整数向量，**a**是一个实数或者复数向量。(u_i, v_i)对应值a_i，如果**a**中有零元素，则将这个元素排除在外。稀疏矩阵的大小为max(u)×max(v)。
`sparse(u,v,a,m,n)`	生成一个m×n的稀疏矩阵，(u_i, v_i)对应值a_i。向量**u**，**v**和**a**必须长度相同。
`sparse(u,v,a,m,n,nzmax)`	生成一个m×n的含有*nzmax*个非零元素的稀疏矩阵。(u_i, v_i)对应值a_i。*nzmax*的值必须大于或者等于向量**u**和**v**的长度。
`find(x)`	返回向量**x**中非零元素的下标。如果 **x**=**X**是一个矩阵，那么**X**的向量就作为一个长向量来考虑。
`[u,v]=find(A)`	返回矩阵**A**中非零元素的下标。
`[u,v,s]=find(A)`	返回矩阵**A**中非零元素的下标。用向量**s**中元素的值及**u**和**v**中相应的下标，实际上就是向量**u**、**v**和**s**作为命令sparse的参数。
`spconvert(D)`	将一个有三列的矩阵转换成一个稀疏矩阵。**D**中的第1列作为行的下标，第2列作为列的下标，最后一列作为元素值。

而且可以使用命令full将稀疏矩阵转换成一个满矩阵。

命令集88 转换成满矩阵

`full(S)`	将稀疏矩阵**S**转换成一个满矩阵。

■ 例9.3

(a) 创建一个5×5的单位矩阵：

```
A=eye(5)
```

将矩阵**A**转换成稀疏矩阵**B**：

```
B = sparse(A)

B =
   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
   (5,5)        1
```

(b) 假设MATLAB中给出如下的向量：

```
ind1 = [1 2 3 3 4 2];
ind2 = [1 2 1 4 5 3];
number = [0 1 2 3 0 5];
```

这样就有了行向量，但是也可使用列向量。运行命令`Smatrix=sparse(ind1,ind2,number)`，结果为：

```
Smatrix =
   (3,1)        2
```

```
(2,2)        1
(2,3)        5
(3,4)        3
```

其中有去掉了两个零元素。将这个矩阵转换成满矩阵，输入：

```
Fullmatrix=full(Smatrix)
```

得到的结果为：

```
Fullmatrix =
     0     0     0     0     0
     0     1     5     0     0
     2     0     0     3     0
     0     0     0     0     0
```

注意，稀疏矩阵和得到的满矩阵的大小是分别是由 **ind1**和**ind2**中最大元素值确定的，即使相应的值是零，并且在列出的稀疏矩阵中去掉这个值。

输入命令whos可得到：

```
  Name            Size      Bytes    Class

  A               5x5         200    double array
  B               5x5          84    sparse array
  Fullmatrix      4x5         160    double array
  Smatrix         4x5          96    sparse array
  ind1            1x6          48    double array
  ind2            1x6          48    double array
  number          1x6          48    double array

Grand total is 74 elements using 684 bytes
```

可以看出虽然两个矩阵的大小相同，但是其中稀疏矩阵需要的存储空间更小些。

(c) 在处理稀疏矩阵时find命令很有用。命令对于稀疏矩阵或者满矩阵都返回相同的结果。返回得到的三个向量直接用来重新创建一个稀疏矩阵。令 Smatrix定义在(b)中，运行命令：

```
[ind1,ind2,number] = find(Smatrix);
Smaller = sparse(ind1,ind2,number)
```

得到的结果为：

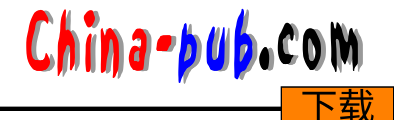
```
Smaller =
   (3,1)        2
   (2,2)        1
   (2,3)        5
   (3,4)        3
```

用下面命令得到的矩阵和 (b)中得到的矩阵是不一样的：

```
Fullsmall = full(Smaller)

Fullsmall =
     0     0     0     0
     0     1     5     0
     2     0     0     3
```

■



9.3 稀疏矩阵运算

MATLAB中对满矩阵的运算和函数同样可用在稀疏矩阵中。结果是稀疏矩阵还是满矩阵，这取决于运算符或者函数及下列的操作数：

• 当函数用一个矩阵作为输入参数，输出参数为一个标量或者一个给定大小的向量时，输出参数的格式总是返回一个满阵形式，如命令 size。

• 当函数用一个标量或者一个向量作为输入参数，输出参数为一个矩阵时，输出参数的格式也总是返回一个满矩阵，如命令eye。还有一些特殊的命令可以得到稀疏矩阵，如命令speye。

• 对于单参数的其他函数来说，通常返回的结果和参数的形式是一样的，如 diag。

• 对于双参数的运算或者函数来说，如果两个参数的形式一样，那么也返回同样形式的结果。在两个参数形式不一样的情况下，除非运算的需要，均以满矩阵的形式给出结果。

• 两个矩阵的组和[A B]，如果**A**或**B**中至少有一个是满矩阵，则得到的结果就是满矩阵。

• 表达式右边的冒号是要求一个参数的运算符，遵守这些运算规则。

• 表达式左边的冒号不改变矩阵的形式。

■ **例**9.4

假设有：

```
A = eye(5); B = sparse(A); h = [1;2;0;4;5];
```

这是一个5×5的单位满矩阵和相应的稀疏矩阵。

(a) C=5*B，结果为：

```
C =
    (1,1)        5
    (2,2)        5
    (3,3)        5
    (4,4)        5
    (5,5)        5
```

这是一个稀疏矩阵。

(b) D=A+B，给出的结果为：

```
D =
      2     0     0     0     0
      0     2     0     0     0
      0     0     2     0     0
      0     0     0     2     0
      0     0     0     0     2
```

这是一个满矩阵。

(c) x=B\h，结果为：

```
x =
      1
      2
      0
      4
      5
```

■

这是一个满向量。

有许多命令可以对非零元素进行操作。

命令集89　矩阵的非零元素

命令	说明
`nnz(A)`	求矩阵**A**中非零元素的个数。它既可求满矩阵也可求稀疏矩阵。
`spy(A)`	画出稀疏矩阵 **A**中非零元素的分布。也可用在满矩阵中，在这种情况下，只给出非零元素的分布。
`spy(A,cstr,size)`	用指定的颜色 **cstr**(见表13-1)和在*size*规定的范围内画出稀疏矩阵**A**中非零元素的分布。
`nonzeros(A)`	按照列的顺序找出矩阵 A中非零的元素。
`spones(A)`	把矩阵**A**中的非零元素全换为1。
`spalloc(m,n, nzmax)`	产生一个$m \times n$阶只有*nzmax*个非零元素的稀疏矩阵。这样可以有效地减少存储空间和提高运算速度。
`nzmax(A)`	给出为矩阵**A**中非零元素分配的内存数。不一定和 `nnz(A)`得到的数相同；参见`sparse`或者`spalloc`。
`issparse(A)`	如果矩阵**A**是稀疏矩阵，则返回1；否则返回0。
`spfun(fcn,A)`	用**A**中所有非零元素对函数 **fcn**求值，如果函数不是对稀疏矩阵定义的，同样也可以求值。
`spfun(A)`	求稀疏矩阵 **A**的结构秩。对于所有的矩阵来说，都有`sprank(A≥rank(A)`。

■ **例**9.5

用下面的命令定义稀疏矩阵：

```
A = sparse(diag(ones(5,1),1)) + sparse(diag(ones(5,1),-1));
```

现在创建一个大矩阵：

```
Big=kron(A, A)
```

这个矩阵**Big**是什么样子呢？Kronecker张量积给出一个大矩阵，它的元素是矩阵 A的元素之间可能的乘积。因为参量都是稀疏矩阵，所以得到的矩阵也是一个稀疏矩阵。可以用命令`whos`和`issparse`来确认一下。

查看矩阵**Big**的结构图，可输入`spy(Big)`，结构如图9-1所示。

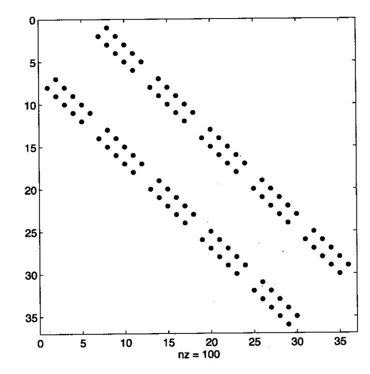
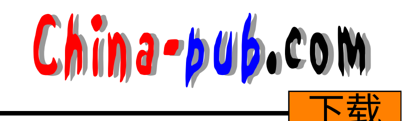


图9-1　用spy命令显示的矩阵结构图



可以看出Big是一个块双对角矩阵。

■

9.4 稀疏矩阵的特例

MATLAB中有四个基本稀疏矩阵，它们是单位矩阵、随机矩阵、对称随机矩阵和对角矩阵。

命令集90　单位稀疏矩阵

`speye(n)`	生成$n\times n$的单位稀疏矩阵。
`speye(m,n)`	生成$m\times n$的单位稀疏矩阵。

命令speye(A) 得到的结果和`sparse(eye(A))`是一样的，但是没有涉及到满阵的存储。

命令集91　随机稀疏矩阵

`sprand(A)`	生成与A有相同结构的随机稀疏矩阵，且元素服从均匀分布。
`sprand(m,n,dens)`	生成一个$m\times n$的服从均匀分布的随机稀疏矩阵，有$dens\times m\times n$个非零元素，$0\le dens\le 1$。参数*dens*是非零元素的分布密度。
`sprand(m,n,dens,rc)`	生成一个近似的条件数为$1/rc$、大小为$m\times n$的随机稀疏矩阵。如果rc=**rc**是一个长度为$1\le l(\min(m, n))$的向量，那么矩阵将rc_i作为它l个奇异值的第一个，其他的奇异值为 0。
`sprandn(A)`	生成与A有相同结构的随机稀疏矩阵，且元素服从正态分布。
`sprandn(m,n,dens,rc)`	生成一个$m\times n$的服从正态分布的随机稀疏矩阵，和sprand一样。
`sprandsym(S)`	生成一个随机对称稀疏矩阵。它的下三角及主对角线部分与S的结构相同，矩阵元素服从正态分布。
`sprandsym(n,dens)`	生成一个$m\times n$的随机对称稀疏矩阵。矩阵元素服从正态分布，分布密度为*dens*。
`sprandsym(n,dens,rc)`	生成一个近似条件数为$1/rc$的随机对称稀疏矩阵。元素以0对称分布，但不是正态分布。如果rc=**rc**是一个向量，则矩阵有特征值rc_i。也就是说，如果**rc**是一个正向量，则矩阵是正定矩阵。
`sprandsym(n,dens,rc,k)`	生成一个正定矩阵。如果k=1，则矩阵是由一正定对称矩阵经随机Jacobi旋转得到的，其条件数正好等于$1/rc$；如果k=2，则矩阵为外积的换位和，其条件数近似等于$1/rc$。
`sprandsym(S,dens,rc,3)`	生成一个与矩阵**S**结构相同的稀疏矩阵，近似条件数为$1/rc$。参数*dens*被忽略，但是这个参数在这个位置以便函数能确认最后两个参数的正确与否。

■ **例9.6**

(a) 假设有矩阵A：

$$\mathbf{A}=\begin{pmatrix}0&1&0&0\\1&0&0&0\\0&1&0&0\\0&0&1&0\end{pmatrix}$$

输入Random=sprandn(A)，可得到随机稀疏矩阵：

```
Random =
    (2,1)      -0.4326
    (1,2)      -1.6656
    (3,2)       0.1253
    (4,3)       0.2877
```

矩阵中随机数的位置和矩阵**A**中非零元素的位置相同。

(b) 对于(a)中的矩阵**A**，输入：

```
B=sprandsym(A)
```

结果为：

```
B =
    (2,1)      -1.1465
    (1,2)      -1.1465
    (3,2)       1.1909
    (2,3)       1.1909
    (4,3)       1.1892
    (3,4)       1.1892
```

这是一个用矩阵**A**的下三角及主对角线部分创建的对称矩阵，在非零元素的位置用随机数作为元素值。

■

用命令spdiags可以取出对角线元素，并创建带状对角矩阵。假设矩阵 **A**的大小为$m\times n$，在p个对角线上有非零元素。**B**的大小为min($m\times n$)$\times p$，它的列是矩阵**A**的对角线。向量**d**的长度为p，其整型分量给定了A的对角元：

d_i<0 主对角线下的对角线。举例如$d_i=-1$，用第一个下对角线

d_i=0 用主对角线

d_i>0 主对角线上的对角线

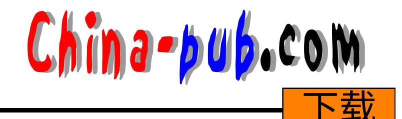
命令集92 对角稀疏矩阵

[B,d]=spdiags(A)	求出**A**中所有的对角元，对角元保存在矩阵 **B**中，它们的下标保存在向量**d**中。
spdiags(A,d)	生成一个矩阵，这个矩阵包含有矩阵**A**中向量**d**规定的对角元。
spdiags(B,d,A)	生成矩阵**A**，用矩阵**B**中的列替换**d**定义的对角元。
A=spdiags(B,d,m,n)	用保存在由**d**定义的**B**中的对角元创建稀疏矩阵**A**。

例11.4给出了如何使用spdiags命令来解普通微分方程组。

9.5 系数阵为稀疏矩阵的线性方程组

在许多实际应用中要保留稀疏矩阵的结构，但是在计算过程中的中间结果会减弱它的稀疏性，如LU分解。这就会导致增加浮点运算次数和存储空间。为了避免这种情况发生，在



MATLAB中用命令对矩阵进行重新安排。这些命令都列在下面的命令集 93中。通过help命令可以得到每个命令更多的帮助信息，也可见 helpdesk。

命令集93 矩阵变换

colmmd(A)	返回一个变换向量，使得矩阵 A列的秩为最小。
symmmd(A)	返回使对称矩阵秩为最小的变换。
symrcm(A)	矩阵A的Cuthill-McKee逆变换。矩阵A的非零元素在主对角线附近。
colperm(A)	返回一个矩阵A的列变换的向量。列按非零元素升序排列。有时这是LU因式分解前有用的变换：lu(A(:, j))。如果A是一个对称矩阵，对行和列进行排序，这有利于Cholesky分解：chol(A(j, j)) 。
randperm(n)	给出正数1，2，…，*n*的随机排列，可以用来创建随机变换矩阵。
dmperm(A)	对矩阵A进行Dulmage-Mendelsohn分解，输入help dmperm可得更多信息。

■ **例**9.7

创建一个秩为4的变换矩阵，可输入：

```
i = [1 2 3 4]; aa = ones(1,4); perm = randperm(4)
P = sparse(i,perm,aa)
```

一旦运行perm=randperm(4)，就会得到：

```
perm =
     4     1     3     2
```

给出的变换矩阵为：

```
P =
   (2,1)        1
   (4,2)        1
   (3,3)        1
   (1,4)        1
```

如果矩阵A为：

$$\mathbf{A}=\begin{pmatrix}7&4&3&4\\5&2&4&2\\5&6&3&1\\8&1&1&2\end{pmatrix}$$

输入命令：

```
RowChange = P*A, ColChange = A*P
```

运行结果为：

```
RowChange =
     5     6     3     1
     5     2     4     2
     8     1     1     2
     7     4     3     4
```

```
ColChange =
     4     4     7     3
     2     2     5     4
     1     6     5     3
     2     1     8     1
```

■

有两个不完全因式分解命令，它们是用来在解大线性方程组前进行预处理的。用helpdesk命令可得更多信息。

命令集94　不完全因式分解

`cholinc(A,opt)`	进行不完全Cholesky分解，变量opt取下列值之一： `droptol` 指定不完全分解的舍入误差，0给出完全分解。 `michol` 如果`michol=1`，则从对角线上抽取出被去掉的元素。 `rdiag` 用`sqrt(droptol*norm(X(:,j)))`代替上三角分解因子中的零元素，*j*为零元素所在的列。
`[L,U,P]= luinc(X,opt)`	返回矩阵**X**的不完全分解得到的三个矩阵**L**、**U**和**P**，变量**opt**取下列值之一： `droptol` 指定分解的舍入误差。 `milu` 改变分解以便从上三角角分解因子中抽取被去掉的列元素。 `udiag` 用`droptol`值代替上三角角分解因子中的对角线上的零元素。 `thresh` 中心极限。

解稀疏线性方程组既可用左除运算符解，也可用一些特殊命令来解。

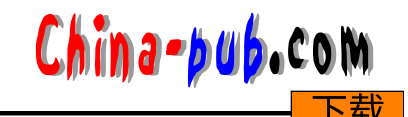
命令集95　稀疏矩阵和线性方程组

`spparms(keystr,op)`	设置稀疏矩阵算法的参数，用 help spparms可得详细信息。
`spaugment(A,c)`	根据[c*I A; A' 0] 创建稀疏矩阵，这是二次线性方程组的最小二乘问题。参见7.7节。
`symbfact(A)`	给出稀疏矩阵的Cholesky和LU因式分解的符号分解因子。用`help symbfact`可得详细信息。

稀疏矩阵的范数计算和普通满矩阵的范数计算有一个重要的区别。稀疏矩阵的欧几里德范数不能直接求得。如果稀疏矩阵是一个小矩阵，则用 `norm(full(A))`来计算它的范数；但是对于大矩阵来说，这样计算是不可能的。然而 MATLAB可以计算出欧几里德范数的近似值，在计算条件数时也是一样。

命令集96　稀疏矩阵的近似欧几里德范数和条件数

`normest(A)`	计算A的近似欧几里德范数，相对误差为 10^{-6}。
`normest(A,tol)`	计算A的近似欧几里德范数，设置相对误差 *tol*，而不用缺省时的10^{-6}。



[nrm,nit]= normest(A)	计算近似*nrm*范数，还给出计算范数迭代的次数*nit*。
condest(A)	求矩阵**A**条件数的1-范数中的下界估计值。
[c,v]= condest(A,tr)	求矩阵**A**的1-范数中条件数的下界估计值c和向量**v**，使得 ||**Av**||=(||**A**|| · ||**v**||)/c。如果给定*tr*，则给出计算的过程。*tr*=1，给出每步过程；*tr*=−1，给出商c/rcond(A)。

■ **例**9.8

假设给出：

```
Sprs = speye(4); Sprs(4,1) = 19; Sprs(3,2) = 4;
```

用normApprox=normest(Sprs)计算出：

```
normApprox =
   19.0525
```

用theNorm=norm(full(Sprs))得：

```
theNorm =
   19.0525
```

为了找到它们之间的差别，计算difference=theNorm-normApprox，得：

```
difference =
   8.5577e-09
```

在许多应用中，normest计算得到的近似值是一个很好的近似欧几里德范数，它的计算步数要比norm要少得多；可参见7.6节。

■

用etree命令来找到稀疏对称矩阵的消元树，用向量 **f**来描述消元树，还可用etreeplot命令画出来。元素f_i是矩阵的上三角Cholesky分解因子中*i*行上第1非零元素的列下标。如果有非零元素，则f_i=0。消元树可以这样来建立：

节点*i*是f_i的孩子，或者如果f_i=0，则节点*i*是树的根节点。

命令集97　矩阵的消元树

etree(A)	求**A**的消元树向量 **f**，这个命令有可选参数；输入 help etree获取帮助。
etreeplot(A)	画出向量**f**定义的消元树图形。
treeplot(p,c,d)	画出指针向量 **p**的树图形，参数*c*和*d*分别指定节点的颜色和分支数。etreeplot可以调用这个命令。
treelayout	显示树的结构，treeplot可以调用这个命令。

■ **例**9.9

假设有对称稀疏矩阵**B**：

$$
\mathbf{B} = \begin{pmatrix} 5 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 5 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 5 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 5 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 5 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 5 \end{pmatrix}
$$

运行命令btree=etree(B)，结果为：

```
btree =
     2     5     5     7     6     7     8     0
```

开始的数字2不难理解，它是矩阵的第1列上第1个非零元素的行数，它决定了在Cholesky分解因子的第1行第2列处有一个非零元素。当缩减第 1列的元素时就得到第 2列的数字5。**B**在缩减后，在(5,2)位置的元素是非零的，这样消元树向量中第2个元素的值为5。spy(chol(B))给出了Cholesky分解因子的结构图，如图9-2所示：

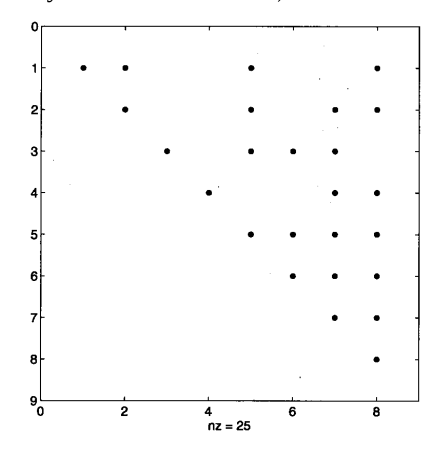


图9-2　Cholesky分解结构图

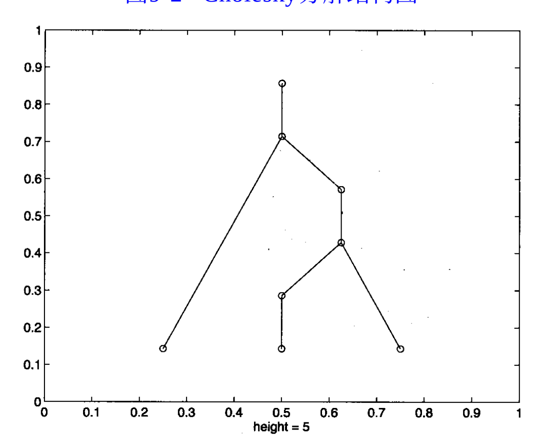
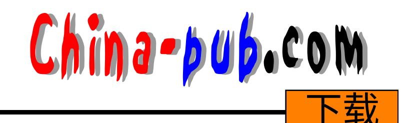


图9-3　矩阵B的消元树

这个向量消元树可以这样来建立：上三角中只有一行有非零元素，节点 8，因此这就是树唯一的根。节点1是节点2的孩子，节点2和3是节点5的孩子，而节点5是节点6的孩子。节点4



和6是节点7的孩子，而节点7又是节点8的孩子，即根的孩子。

命令etreeplot(B)给出了树的结构图，如图9-3所示。

消元树的形状取决于列和行序，它可以用来分析消元过程。

■

用gplot命令可以画出坐标和矩阵元素间的联系图形。必须在$n \times 2$的矩阵中给出n个坐标，矩阵的每一行作为一个点。这样就创建出点点之间连接的 $n \times n$矩阵，如果点4连接到点8，则(4, 8)的值为1。由于是一个大矩阵，而且非零元素较少，所以它应该被建成稀疏矩阵。

这个图可以说明网络问题，如传递问题。它还包含有线性方程组中未知量之间的相关信息。

命令集98　网络图形

gplot(A,K)	如果矩阵**A**的$a(i, j)$不为0，则将点k_i连接到点kj。**K**是一个$n \times 2$的坐标矩阵，**A**是一个$n \times n$的关联矩阵。
gplot(A,K,str)	用字符串str给定的颜色和线型画出的同上图形。字符串 **str**的取值参见表13-1。
[X,A]=unmesh(E)	求网格边界矩阵**E**的Laplace矩阵**A**和网格点的坐标矩阵**X**。

■ **例**9.10

假设有下面的坐标矩阵**K**和关联矩阵**A**：

$$\mathbf{K} = \begin{pmatrix} 0 & 1 \\ 1 & 0.2 \\ 1.3 & 0.9 \\ 2 & 0 \\ 2 & 1.9 \\ 3 & 2 \\ 4 & 1 \end{pmatrix} \qquad \mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

矩阵A在稀疏化后，用命令gplot(A,K)画出图9-4，给出了点(0, 1)和点(4, 1)之间所有可能的路径。

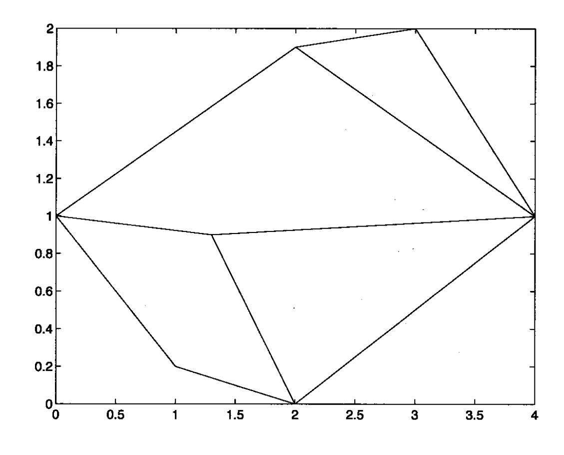


图9-4　使用gplot的一个例子

■



第10章 函数、插值和曲线拟合分析

MATLAB可以处理有估值和没有估值的多项式，还有一些强大的数值分析命令，如求零值和最小值。MATLAB中还有数据集合的插值、曲线拟合的命令和函数，还提到了经典的贝赛尔(Bessel)函数。

10.1 MATLAB中的多项式

MATLAB将阶为n的多项式$p(x)$存储在长度为$n+1$的行向量**p**中。元素为多项式的系数，并按x的幂降序排列，表示为：

$$\mathbf{p} = (a_n \quad a_{n-1} \quad \dots \quad a_1 \quad a_0)$$

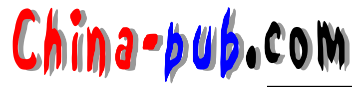
代表的多项式为：

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

令**A**是一个稀疏矩阵，向量**p**和**q**的长度分别为$n+1$和$m+1$，它们分别表示次数为n和m的多项式。MATLAB中处理多项式的命令如下：

命令集99　多项式

命令	说明
`polyval(p,x)`	计算多项式**p**。如果x是一个标量，则计算出多项式在x点的值；如果**x**是一个向量或者一个矩阵，则计算出多项式在**x**中所有元素上的值。
`[y,err]=polyval(p,x,E)`	计算向量**x**的多项式**p**的值。同上，计算结果在**y**中，同时还根据`polyfit`命令给出的矩阵**E**返回一个误差估计向量**err**。见`help polyval`和`help polyfit`，参见10.4节。
`polyvalm(p,A)`	直接对矩阵**A**进行多项式计算。不是象上个命令一样对每个元素进行多项式计算，而是计算$p(\mathbf{A})=p_1\mathbf{A}^n+p_2\mathbf{A}^{n-1}+\cdots+p_{n+1}\mathbf{I}$。
`poly(A)`	计算矩阵**A**的特征多项式向量。
`poly(x)`	给出一个长度为$n+1$的向量，其中的元素是次数为n的多项式的系数。这个多项式的根是长度为n的向量**x**中元素。
`compan(p)`	计算带有系数p的多项式的友矩阵**A**，这个矩阵的特征多项式为**p**。
`roots(p)`	计算特征多项式**p**的根，是一个长度为n的向量，也就是方程$p(x)=0$的解。表达式`poly(roots(p))=p`为真。结果可以是复数。
`conv(p,q)`	计算多项式**p**和**q**的乘积，也可以认为是**p**和**q**的卷积。
`[k,r]=deconv(p,q)`	计算多项式**p**除**q**。**k**是商多项式，**r**是残数多项式。这个计算等价于**p**和**q**的逆卷积。



[u v k]= residue(p,q)	计算$p(x)/q(x)$的部分展开式： $$\frac{p(x)}{q(x)}=\frac{u(1)}{x-n(1)}+\frac{u(2)}{x-n(2)}+\frac{u(j)}{x-n(j)}+k(x).$$ 向量**p**和**q**分别是多项式$p(x)$和$q(x)$的系数。得到的余数在向量**u**中，极点在列向量**v**中，商多项式在向量**k**中。
[p q]=residue(u,v,x)	从同上的部分展开式**u**、**v**和**x**计算得到多项式**p**和**q**。
mpoles	给出极点多样性的相关信息，见help mpoles
polyder(p)	计算得到长度为n的微分多项式向量，多项式的系数在向量**p**中。
polyder(p,q)	返回一个向量，它表示由conv(p,q)定义的多项式微分。
[u,v]=polyder(p,q)	返回两个向量，它们表示由deconv(p,q)定义的多项式微分，表达形式为**u**/**v**。

■ **例**10.1

将一些多项式命令应用在下列的多项式上：

$p2(x)=3x^2+2x-4 \qquad p3(x)=2x^3-2$

MATLAB用下列向量来表示这两个多项式：

```
p2 = [3 2 -4];
p3 = [2 0 0 -2];
```

(a) 计算多项式在x=1处的值，输入：

```
value2 = polyval(p2,1), ...
value3 = polyval(p3,1)
```

结果为：

```
value2 =
     1

value3 =
     0
```

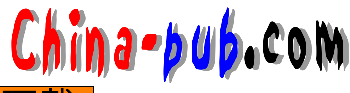
(b) 很容易对向量或者矩阵计算多项式的值，输入：

```
x = [1 2 3]';
values2 = polyval(p2,x), values3 = polyval(p3,x)
```

结果为：

```
values2 =
     1
    12
    29

values3 =
     0
    14
    52
```



(c) 两个多项式相乘，得到一个新的多项式：

```
p5=conv(p2, p3)
```

给出：

```
p5 =
      6      4     -8     -6     -4      8
```

(d) 用roots命令求多项式的根：

```
roots2=roots(p2), roots3=roots(p3)
```

给出：

```
roots2 =
   -1.5352
    0.8685

roots3 =
  -0.5000 + 0.8660i
  -0.5000 - 0.8660i
   1.0000
```

图10-1中显示出两个多项式的图形。

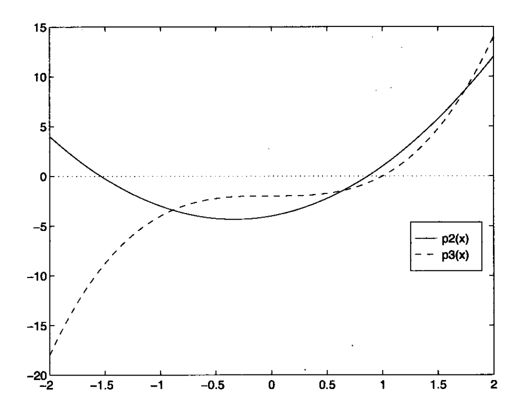


图10-1　多项式$p2(x)=3x^2+2x-4$和$p3(x)=2x^3-2$

(e) 多项式$p(x)$的牛顿-拉普森迭代，多项式的系数在向量**p**中：

```
q = polyder(p);
xnext = x - polyval(p,x)/polyval(q,x);
```

(f) 命令roots(poly(A))求得矩阵**A**的特征值。假设矩阵**A**为：

$$\mathbf{A} = \begin{pmatrix} -9 & -3 & -16 \\ 13 & 7 & 16 \\ 3 & 3 & 10 \end{pmatrix}$$

运行命令：

```
usedRoots=roots(poly(A))
```

结果为：

```
usedRoots =
   10.0000
    4.0000
   -6.0000
```

然而这样得到的特征值没有用 MATLAB命令eig(A)得到的特征值的精度高，而且有效性也差些：

```
usedEig=eig(A)
```

给出：

```
usedEig =
  -6.0000
  10.0000
   4.0000
```

结果是一样的，但是顺序正好相反。

(g) 对于所有的矩阵**A**都有：polyvalm(poly(A)， A)=0

这是Cayley-Hamilton法则。这个法则对于秩为5的方阵来说：

```
Magical = magic(5);
AlmostZero = polyvalm(poly(Magical),Magical)

AlmostZero =
  1.0e-07 *
   0.2794    0.3551    0.1723    0.1770    0.2654
   0.2765    0.2887    0.2049    0.2142    0.2561
   0.1775    0.2468    0.2701    0.3073    0.2375
   0.1942    0.2744    0.2759    0.2608    0.2282
   0.2082    0.3120    0.2608    0.2515    0.2049
```

■

10.2 函数的零值

MATLAB的M文件可以表示数学函数；参见2.9节。函数：

$$g(x)=\frac{5x-6.4}{(x-1.3)^2+0.002}+\frac{9x}{x^3+0.03}-\frac{x-0.4}{(x-0.92)^2+0.005}$$

如果输入下面的M文件g.m，这个函数就可以在MATLAB中调用：

```
function y = g(x)

y = (5.*x-6.4)./((x-1.3).^2+0.002) + ...
    (9.*x)./(x.^3+0.03) - ...
    (x-0.4)./((x-0.92).^2+0.005);
```

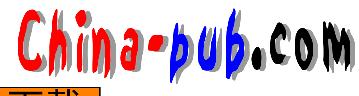
使用元素运算符.*、./、.^、+和一定义MATLAB函数**g**。结果是如果这个函数被一个向量调用，那么得到的结果也是一个向量。本章中提到的所有MATLAB函数需要以这种方式来定义数学函数。

用plot命令可以画出函数的图形：

```
x=linspace(0, 2);            % 生成向量x
plot(x,g(x));                % 画g(x)图形
grid;                        % 画格栅
title('The g(x) function')   % 给出图标题
```

或者使用fplot命令：

```
fplot('g', [0 2]);           % 画g(x)图形
grid;                        % 画格栅
title('The g(x) function')   % 给出图标题
```



结果如图10-2所示。命令plot和fplot都定义在13.1 节中。

求函数 $f(x)$ 的零值就等于求方程 $f(x)=0$ 的解。单变量函数的零值可以用 MATLAB命令fzero来求。对于多项式可以用 roots命令来求，参见10.1节。fzero用迭代法来求解，使得初始的估计值接近理想的零值。

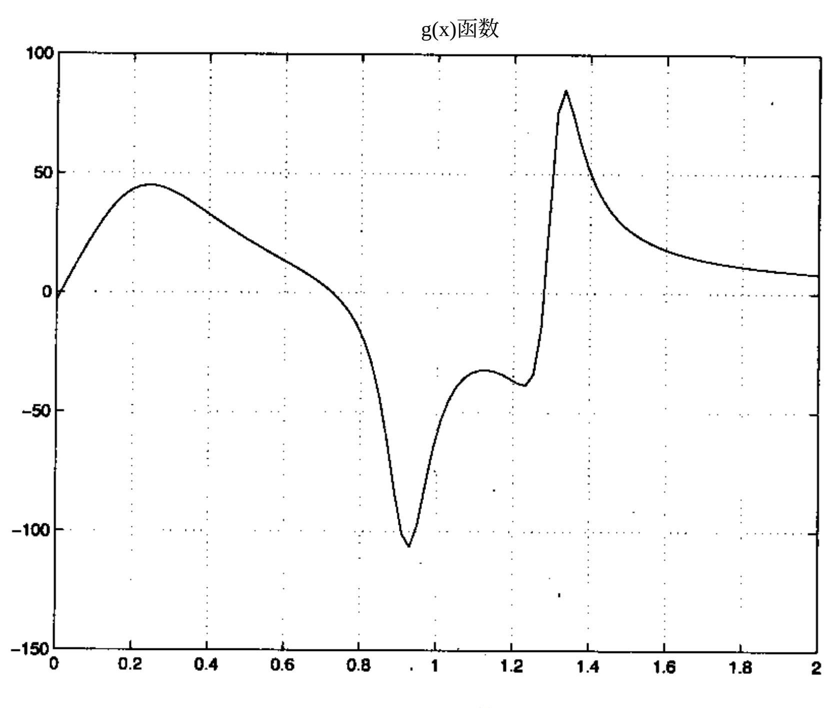


图10-2　用fplot 画的 $g(x)$ 图形

命令集100　函数的零值

fzero(fcn,x0)	求函数的一个零值， **fcn**为函数的名字。要求给出一个初始值x0，近似值的相对误差为eps。
fzero(fcn,x0,tol)	求函数的一个零值， **fcn**为函数的名字。要求给出一个初始值x0，由用户定义近似值的相对误差 *tol*。
fzero(fcn,x0,tol,pic)	求函数的一个零值，同上。如果pic不为零，则给出迭代过程。
fzero(fcn,x0,tol, pic p1 p2,...)	求多变量函数的零值，**fcn=fcn**(x0, p1, p2,…)。如果tol和pic没有给出，则令它们为空矩阵，如 fzero(fcn,x0,[],[],p1)。

zerodemo命令给出了一个演示实列。

■ **例**10.2

(a) 求本节开头定义的函数 $g(x)$ 的零值：

```
x1 = fzero('g',0), x2 = fzero('g',0.5), x3 = fzero('g',2)
```

结果为：

```
x1 =
    0.0112

x2 =
    0.7248

x3 =
    1.2805
```

(b) 求函数sin$x$和2x—2的交集，也就是求方程 sin x=2x-2的解。先定义函数 *sinm(x)*，将它存放在M文件sinm.m中，如下：

```
function s = sinm(x)


s = sin(x) - 2.*x + 2;
```

画出曲线是找到初始值的一个好方法，所以：

```
fplot('sinm',[-10 10]);
grid on;
title('The sin(x) - 2.*x + 2 function');
```

结果如图10-3所示。可以看出2是一个可接受的估计值，输入：

```
xzero=fzero('sinm', 2)
```

结果为：

```
xzero =
   1.4987
```

这就是方程sin$x$=2x—2的解。

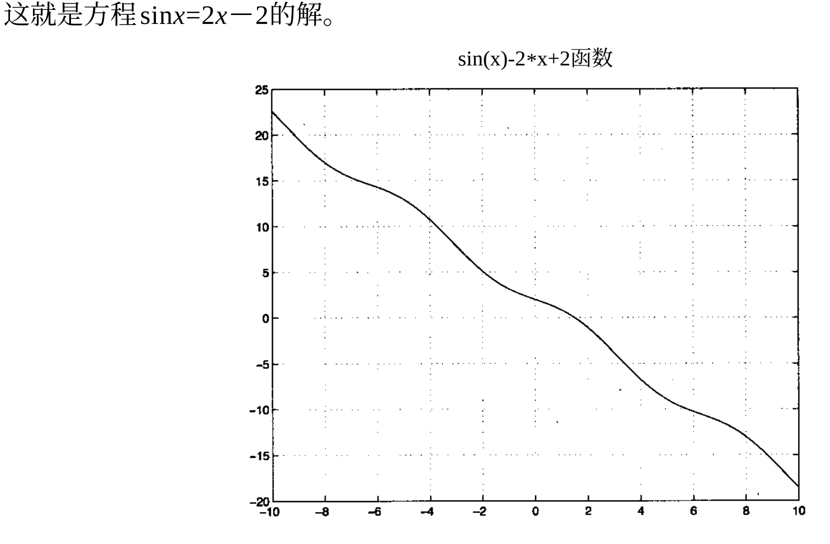


图10-3 *sinm*(*x*)函数

10.3 函数的最小值和最大值

最优化是求最优解，也就是在某个区间内有条件约束或者无条件约束地找到函数的最大值或者最小值。MATLAB使用数字方法求函数的最小值。使用迭代算法，也就是有些步骤要重复许多次。现在，假设要求函数 f在某个区间内的最小值 x_{min}。

$$f(x_{\min}) = \min_x f(x)$$

迭代方法需要一个初始估计值 x_0。从x_0开始找到一个更接近x_{min}的新值x_1，这个值的好坏取决于使用的数学方法。直到找到有足够精度的近似值 x_i才停止迭代，也就是绝对值 $|x_{min}-x_i|$足够小。

如果函数有多个局部最小值，fmin可以找到它们中的一个。也可用 MATLAB的最优化工具箱来求得，参见附录C。

这里提到了标准 MATLAB系统的两个最优化命令，fmin命令可以求单变量函数的最小值；fmins命令可以求多变量函数的最小值，同时它还要求有一个初始向量。

没有求函数f的最大值的命令，相反函数 $h=-f$ 的最小值可以求得。

命令集101　函数的最小值

fmin(fcn,x1,x2)	求函数在区间($x1$，$x2$)内的最小值，**fcn**是目标函数名。如果没有局部最小值，则返回区间内的最小 x值。相对误差小于10^{-4}。
fmin(fcn,x1,x2, options)	求函数在区间($x1$，$x2$)内的最小值，**fcn**是目标函数名。如果没有局部最小值，则返回区间内的最小 x值。向量**options**为控制参数，如 **options**(1)=1，显示中间结果；**options**(2)表示得到的结果 x的精度，缺省为 10^{-4}。输入help foptions可得更多信息。
fmins(fcn,x0)	求函数**fcn**的最小值。由用户自己给出一个初始估计向量**x0**，相对误差为10^{-4}。
fmins(fcn,x0, options)	带优化参数求函数**fcn**的最小值，同上。输入help fmins 和 help foptions可得更多信息。例如，优化参数可以控制迭代次数和计算结果的精度。

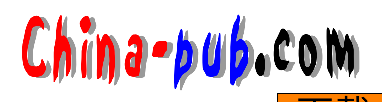
■ **例**10.3

(a) 在区间[0，2π]内求函数cos的最小值：

```
cosmin=fmin('cos', 0,2*pi)    %求cos的最小值
cosmin=
        3.1416
```

这就是期望得到的结果。

(b) 同样可以简单地求高级函数的最小值。对定义在 10.2节中的函数$g(x)$，求在区间[0，2]内的最小值。




```
gmin = fmin('g',0,2)

gmin =
    1.2277
```

注意，这是一个局部最小值，不一定是函数在这个区间内的最小值。从图 10-2上可以看出在一个更小区间内可以得到第二个最小值，这个值比第一个值还小：

```
gmin2 = fmin('g',0,1)

gmin2 =
    0.9260
```

(c) 还可以用fmin命令来求函数的最大值，但是要先编写一个返回－ $g(x)$的函数，这个函数保存在M文件minusg.m中。

```
function y = minusg(x)

y = -g(x);
```

求这个函数的最小值就等于求函数 g的最大值。

```
gmax=fmin('minusg', 0, 2)
```

结果为：

```
gmax =
    0.2433
```

在这个区间内有若干个最大值， MATLAB求出的最大值不一定是函数的全局最大值。

(d) 用fmins命令来求多个变量函数的最小值，假设函数为：

$$f(x_1, x_2) = x_1^2 + x_2^2 - 0.5x_1x_2 - \sin x_1$$

编写M文件fx1x2.m：

```
function f = fx1x2(x)

f = x(1).^2 + x(2).^2 - 0.5.*x(1).*x(2) - sin(x(1));
```

函数fmins要求有一个初始估计向量，假设给 (1， 0)：

```
fx1x2min = fmins('fx1x2',[1,0])
```

结果为：

```
fx1x2min =
    0.4744    0.1186
```

用下面的程序画出函数的图形来：

```
x=linspace(－1, 1 50);      % 新建向量x，假设y=x
for i=1: 50                 % 计算fx1x2在每一点的值
     for j=1: 50
       Z(i, j)=fx1x2([x(i)  x(j)]);
     end
  end
meshc(x ,x, Z);             % 带有基本等值线的网格图
view(80, 10);               % 指定观察点
```

命令meshc画出函数的表面图形，同时在*xy*平面画出图形的等值线。命令meshc和view定义在13.5节中，在4.2节中提到了命令linspace。结果如图10-4所示，从图上可以看出最小值。

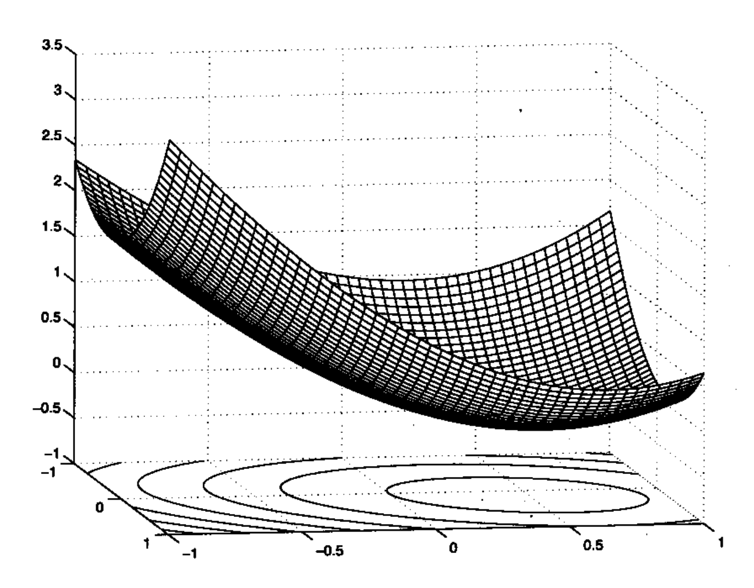


图10-4 函数$x_1^2+x_2^2-0.5x_1x_2-\sin x_1$在区间[−1，1]×[−1，1]上的图形

■

10.4 插值、曲线拟合和曲面拟合

如果在有限个数据点内给出函数，那么利用插值的方法就可以找到中间点的近似值。最简单的插值就是对两个相邻数据点进行线性插值。 interp1和interp2命令用特殊的算法来进行等距离数据点的快速插值。使用时，必须在方法的名字前加上一个星号， ’*’，如interp1(x, Y, xx ,’*cubic’) 。

MATLAB中有几个函数可以用不同的方法来进行数据插值。

命令集102 插值

interp1(x,y,xx)	返回一个长度和向量**xx**相同的向量**f(xx)**。函数**f**由向量**x**和**y**定义，形式为 **y**=**f(x)**，用线性插值的方法来计算值。为了得到正确的结果，向量 **x** 必须按升序或降序排列。
interp1(x,Y,xx)	返回一个相应向量的矩阵 **F(xx)**，同上。矩阵 **Y** 的每列是一个关于**x**的函数，对于每个这样的函数**xx**的值通过插值得到。矩阵 **F(xx)**的行数和向量**xx**的长度相同，列数和矩阵 **Y**的相同。
interp1(x,y,xx, metstr)	进行一维插值，字符串**metstr**规定不同的插值方法，可用的方法有：



	‘linear’ 线性插值。 ‘nearest’ 最邻近插值。 ‘spline’ 三次样条插值。也叫外推法。 ‘cubic’ 三次插值，要求x的值等距离。 所有插值方法均要求x是单调的。
`interp1q(x,y,xx)`	和interp1相同，但是对于非均匀空间的数据插值更快。
`interp2(X,Y,Z,Xx,Yy)`	进行矩阵**Xx**和**Yy**的二维插值，并由**X**、**Y**和**Z**所描述的函数**Z**=**f**(*X*,*Y*)内的插值所决定。如果**X**、**Y**和**Z**中任何一个是一个向量，则它的元素被认为应用于相应的行和列。
`interp2(X,Y,Z,Xx,Yy, metstr)`	进行二维插值，字符串**metstr**规定了不同的插值方法，可用的方法有： ‘linear’ 线性插值。 ‘nearest’ 最邻近插值。 ‘spline’ 三次样条插值。 ‘cubic’ 三次插值。
`VV=interp3(X,Y,Z,V,XX,YY,ZZ,metstr)`	进行由**X**，**Y**和**Z**所描述的函数**V**的插值，**XX**、**YY**和**ZZ**是插值点。字符串**metstr**规定了不同的插值方法，可用的方法有： ‘nearest’ 使用最邻近点的值。 ‘linear’ 使用8个最邻近点进行线性插值。 ‘spline’ 三次样条插值。 ‘cubic’ 使用64个最邻近点进行三次插值。
`VV=interpn(X1,X2,X3, ···,V,Y1,, Y2,Y3,···,method)`	和interp3相同，但是V和VV可以是多维数组。如果**X**1，**X**2，**X**3，…是等距离的，使用星号*如’*cubic’可以加快计算速度。
`Interpft(y,n)`	快速傅利叶变换插值，返回一个长度为*n*，从*y*计算得到的向量。要求*y*的值是等距离的，结果在与*y*相同区间内计算。
`griddata(x,y,z,Xx,Yy,’method’)`	返回相同大小的矩阵**Xx**和**Yy**，它们表示一个网格，由函数*z*=*f*(*x*, *y*)的插值得到。向量**x**、**y**和**z**包含的是三维空间的 *x*、*y*和*z*坐标。字符串**method**规定了不同的插值方法，可用的方法如下： ‘linear’ 基于三角的线性插值。 ‘nearest’ 最邻近插值。

	‘cubic’ 基于三角的三次插值。
	‘v4’ 使用MATLAB 4的插值方法。
[X1,X2,X3,···]= ndgrid(x1,x2,x3, ···)	变换由向量**x1**，**x2**，**x3**···给出的域。对于矩阵**X1**，**X2**，**X3**，···来说，可以用做多变量函数的估计值和多维插值。矩阵 **Xn**的第n维和向量**xn**的元素相同。
[X1X2,···] =ndgrid(x)	等同于[X1,X2,···]= ndgrid(x,x,x,···)。

■ **例**10.4

做一个函数$\sin x^2$在区间[0，2π]上40个函数值的表。

```
x = linspace(0,2*pi,40); y = sin(x.^2);
```

(a) 用interp1来计算中间点的$\sin x^2$函数值，而不用sin求，命令为：

```
values = interp1(x,y,[0 pi/2 3])
```

结果为：

```
values =
        0      0.6050      0.3559
```

和用sin计算得到的正确结果比较：

```
correct = sin([0 pi/3 3].^2)

correct =
        0      0.8897      0.4121
```

在表中用更多的数据点可以使精度更好。

(b) 用样条插值可得更高精度的结果。假设向量 x和y定义如上，那么：

```
better = interp1(x,y,[0 pi/2 3],'spline')
```

结果为：

```
better =
        0      0.6241      0.4098
```

这是一个更好的近似值。

■

命令griddata可以在三维空间内建立任意数据点外的函数。

■ **例**10.5

先生成三个有10个元素的向量，它们的值随机分布在 0～1之间：

```
x = rand(10,1); y = rand(10,1); z = rand(10,1);
```

建立一个网格来计算内部曲面之后，可以用命令 griddata来对这些点之间的曲面进行插值。下面的图10-5给出了不同的插值方法之间的区别：

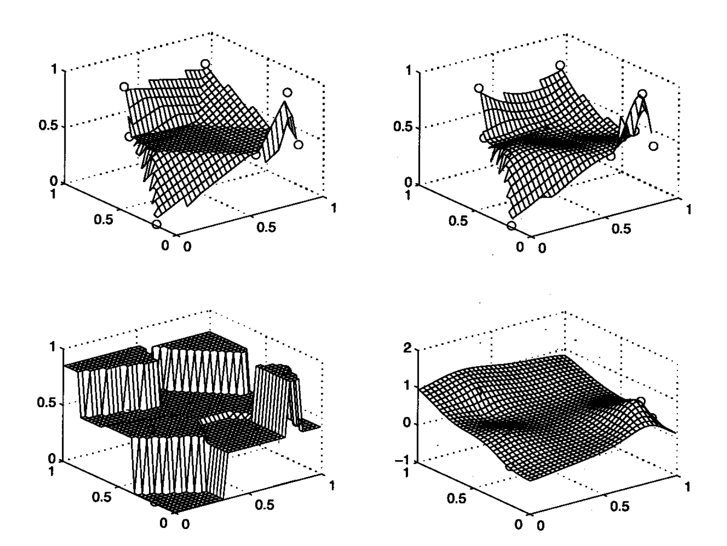


图10-5 用griddata 对10个随机点的插值，左上图用的是‘linear ’，右上图用的是‘cubic ’，右下图用的是‘nearest ’，右下图用的是‘v4’

```
stps=0:0.03:1;                              % 向量A的值在[0, 1]之间
[X,Y]=meshgrid(stps);                       % 生成一个[0,1]×[0,1]坐标网格
Z1=griddata(x, y, z, X, Y);                 % 线性插值
Z2=griddata(x, y, z, X, 'cubic');           % 三次插值
Z3=griddata(x, y, z, X, 'nearest');         % 最邻近插值
Z4=griddata(x, y, z, X, 'v4');              % MATLAB 4中的插值方法
subplot(2, 2, 1);                           % 画第1个子图
mesh(X, Y, Z1);                             % 画曲面网格
hold on                                     % 保持当前图形
plot3(x, y, z,'o');                         % 画出数据点
hold off                                    % 释放图形
subplot(2 2 2);                             % 画第2个子图
mesh(X, Y ,Z2);                             % 画曲面网格
hold on                                     % 保持当前图形
plot3(x, y, z,'o');                         % 画出数据点
hold off                                    % 释放图形
subplot(2, 2, 3);                           % 画第3个子图
mesh(X, Y, Z3);                             % 画曲面网格
hold on;
plot3(x, y, z,'o');                         % 画出数据点
hold off
subplot(2, 2, 4);                           % 画第4个子图
mesh(X, Y, Z4);                             % 画曲面网格
hold on
plot3(x, y, z,'o');                         % 画出数据点
hold off
```

命令hold和subplot在13.3节进行了说明，命令meshgrid在13.4节中，命令mesh和plot3可以在13.5节中找到。结果如图10-5所示。

■

用spline命令可以求得三次样条的近似值，还可以求得三次样条插值 pp形式的向量，向量的元素是三次样条函数的系数。用 ppval命令可以求得三次样条函数的估计值。

命令集103　三次样条数据插值

spline(x,y,xx)	等同于interp1(x, y, xx, 'spline')，但是参数必须是向量。
spline(x,y)	返回三次样条插值向量的 **pp形式**，它是函数 $y=f(x)$的近似值。pp是 ' piecewise polynomial ' 的缩写，得到的向量元素包含计算的三次样条系数。这个命令可以被 ppval函数使用。
YI=splncore(X,Y,XI)	在细胞数组**X**的函数**Y**内进行n维数据的插值，得到一个新集合**XI**。函数可以被interp2、interp3和interpn使用。
ppval(pp,xx)	计算三次样条函数。如果三次样条定义为pp=spline(x, y)，那么ppval(pp,xx)得到的结果和spline(x,y,xx)一样。
p=mkpp(points, coeff,d)	通过用给定点建立 pp函数来求分段多项式，其中 **coeff**(i, :)包含第i个多项式的系数。多项式的个数由 l=length(points)－1确定，第i个多项式的阶数为n=length(coeff(:))/l。
[points,coeff,l, n,d]=unmkpp(p)	给出分段多项式相关信息，见上。

多项式可以使用最小二乘法来进行数据拟合 (也可见7.7节)，用的命令是polyfit。

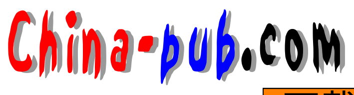
命令集104　多项式曲线拟合

polyfit(x,y,n)	找到次数为n的多项式系数，对于数据集合$\{(x_i, y_i)\}$，满足差的平方和最小。
[p,E]=polyfit(x,y,n)	返回同上的多项式 **P**和矩阵**E**。多项式系数在向量 **p**中，矩阵**E**用在polyval函数中来计算误差。

■ **例10.6**

下面给出了对向量**x**和**y**拟合不同阶的多项式图形，阶分别为3，4，5。

```
x = [-3 -1 0 2 5.5 7];
y = [3.3 4.5 2.0 1.5 2.5 -1.2];
p3=polyfit(x, y, 3);            % 用向量x和y中元素拟合不同
p4=polyfit(x, y, 4);            % 次数的多项式
p5=polyfit(x, y, 5);
xcurve= -3.5:0.1:7.2;           % 生成x值
p3curve=polyval(p3, xcurve);    % 计算在这些x点的多项式值
p4curve=polyval(p4, xcurve);
p5curve=polyval(p5, xcurve);
plot(xcurve,p3curve,'--',xcurve,p4curve,'-.', ...
    xcurve,p5curve,'-',x,y,'*');
```




```
lx = [-1 1.5]; ly = [0 0]; hold on;

plot(lx,ly,'--',lx,ly-1.3,'-.',lx,ly-2.6,'-');

text(2,   0,'degree 3');
text(2,-1.3,'degree 4');
text(2,-2.6,'degree 5');
hold off;
```

结果如图10-6所示。

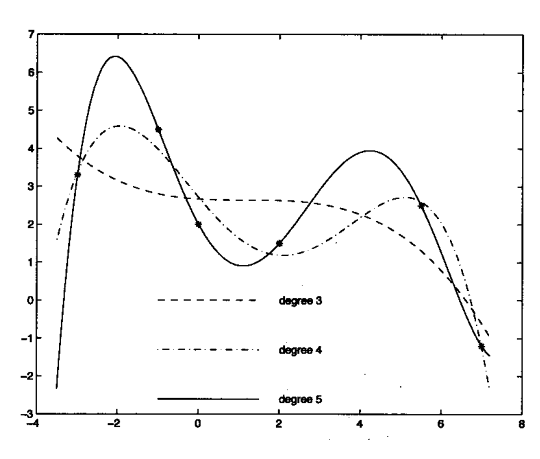


图10-6　不同次数的多项式拟合曲线图

可以看出，次数越高的多项式精度越好。 5次的多项式经过所有的6个点。对于这个特殊的数据集合来说，这个多项式可称为内部插值多项式。

■

在MATLAB中，用legendre命令来计算标量或者向量的**勒让德函数**，它是在选定的区间内形成完全直交集合的直交多项式系统。**勒让德多项式**是阶为零的勒让德函数，可以在给定的数据集合内进行曲线拟合。**贝塞尔函数**是一个经典的特殊函数，它用在数学物理学中。

命令集105　勒让德函数和贝塞尔函数

`legendre(n,x)`	返回一个矩阵，它的元素是在 **x**内计算得到的 *n*阶，*m*=0,1，…，*n*的相关勒让德函数值。**x**的元素要求在区间[−1，1]内。矩阵的第1行对应*m*=0，并包含根据x计算得到的*n*阶勒让德多项式。
`besselj(order,z)`	计算第1类贝塞尔函数，变量 *order*指定函数的阶， **z**用来进行函数运算。
`bessely(n,x)`	计算第2类贝塞尔函数，变量 *order*指定函数的阶， **z**用来进行函数运算。
`besselh(order,k,z)`	计算向量**z**中元素的Hankel函数值(第3类贝塞尔函数)，变量*k*定义使用哪一类Hankel函数。
`besseli(order,z)`	计算改良型第1类贝塞尔函数，变量 *order*指定函数的阶，**z**用来进行函数运算。

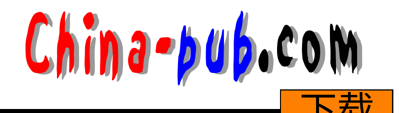
`besselk(order,z)`	计算改良型第2类贝塞尔函数，变量 *order*指定函数的阶，**z**用来进行函数运算。
`w=airy(k,z)`	*k*=0或者不给出*k*，计算Airy函数w=Ai(z)； *k*=1，计算导数Ai´(z)；*k*=2，计算第2类Airy函数Bi(z)； *k*=3，计算导数Bi´(z)。
`[w,err]=airy(...)`	返回一个错误标志数组*err*。

10.5　信号分析

这里简短地介绍一些MATLAB中用作信号分析的命令。通过help和demo命令可以得到更多信息，也可参见‘信号过程工具箱’和它的手册。复数命令(2.4节)和卷积命令(10.1节)也要涉及。

命令集106　信号分析

`fft(x)`	进行向量**x**的离散傅立叶变换。如果 **x**的长度是2的幂，则用快速傅立叶变换， FFT。注意，变换没有规格化。
`fft(x,n)`	得到一个长度为*n*的向量。它的元素是**x**中前*n*个元素离散傅立叶变换值。如果**x**有*m*<*n*个元素，则令最后的*m*+1，…，*n*元素等于零。
`fft(A)`	求矩阵**A**的列离散傅立叶变换矩阵。
`fft(A,n,dim)`	求多维数组*A*中*dim*维内列离散傅立叶变换矩阵。
`ifft(x)`	求向量**x**的离散逆傅立叶变换。用因子1/*n*进行规格化，*n*为向量的长度。也可象fft命令一样对矩阵或者固定长度的向量进行变换。
`fft2(A)`	求矩阵**A**的二维离散傅立叶变换矩阵。这个矩阵没有正交化。如果**A**=**a**是一个向量，则这个命令等同于 fft(a)命令。
`fft2(A,m,n)`	求矩阵**A**中相应元素的二维离散傅立叶变换矩阵。这个矩阵大小为*m*×*n*，没有正交化。如果**A**是一个小矩阵，则余下的元素用零补上。如果可能，MATLAB在计算时用快速傅立叶变换，即FFT。
`ifft2(A)`	求矩阵**A**的二维离散逆傅立叶变换矩阵。用因子1/*mn*进行规格化。可以象ff2一样改变变换矩阵的维数。
`fftn(A,Size)`	求*n*维数组**A**的*n*维离散傅立叶变换数组。如果 **A**是一个向量，则得到的结果也一个向量。如果给定了数组的大小 **Size**，则**X**重构成与**Size**一样大小。
`ifftn(A,Size)`	求*n*维数组**A**的*n*维离散逆傅立叶变换数组。如果 **A**是一个向量，则得到的结果也一个向量。如果给定了数组的大小 **Size**，则**X**重构成与**Size**一样大小。
`fftshift(A)`	返回一个将矩阵**A**的第1象限和第3象限、第2象限和第4象限互换的数组。如果**A**是一个向量，则返回一个左半部和右半部互换的向量。如果**A**是一个*n*维数组(*n*>2)，则返回一个同等大小的数组，将**A**的每一维内的左右半部互换。



ifftshift(A)	fftshift(A)命令的逆变换。
filter(b,a,x)	由向量**a**和**b**所描述形成的滤波器对x向量进行数字滤波，产生滤波后的数据。输入help filter可得更多信息。
Y=filter2(h,X)	用在矩阵**h**中的**FIR**滤波器处理**x**中的数据。结果**Y**由二维卷积计算得到，包含卷记的中心部分，且与**X**的大小相同。
Y=filter2(h, X,form)	Y由二维卷积计算得出，其维数由参数**form**规定，*form*是下列字符串之一： 'full'　返回二维卷积的全部，这样**Y**就比**X**大。 'same'　等同于Y=filter2(h,X)。 'valid'　仅仅返回卷积的一部分，这部分是不带零插值边缘计算的。这样**Y**就比**X**小。

■ **例**10.7

新建一个名为'hat funtion'的函数，之后对其进行傅立叶变换。这个函数在 0,1处值为0，在0.5处值为1。

用linspace建立这个函数：

```
x = linspace(0,1,100);
y = [linspace(0,1,50) linspace(1,0,50)];
```

用下列命令画出函数的图形，如图 10-7所示：

```
subplot(1,3,1); plot(x,y);
title('A hat function');
```

进行傅立叶变换：

```
subplot(1,3,2); plot(x,fft(y));
title('The Fourier transform');
```

这个傅立叶变换得到复数值，但是只显示出实数部分。最后，对它进行逆傅立叶变换得到原来函数：

```
subplot(1,3,3); plot(x,ifft(fft(y)));
title('Retransformed hat function');
```

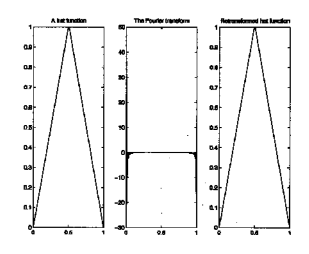


图10-7　函数的傅立叶变换图

所有的画图命令都定义在第13章中。

■



第11章　积分和微分方程组

在有效的MATLAB命令帮助下，可以求解出定积分和普通微分方程的数字解并绘制出其图形。

11.1　积分

在MATLAB中能求解如下形式的定积分并给出数字解：

$$q = \int_a^b g(x)\,dx$$

有许多方法都可以能够解决积分问题（又叫做求面积）。如果要用MATLAB监控整个计算过程，可以使用quad命令。同样能计算出被积函数g的值，并且让MATLAB使用梯形规则和trapz命令计算出积分。当只有离散的数据点和被积函数的数学表达式为未知时，这种方法是非常有效的。

命令集107　定积分计算

命令	说明
`trapz(x,y)`	计算出函数x的积分并将结果返回到y。向量*x*和*y*有相同的长度，*(xi, yi)*代表曲线上的一点。曲线上点的距离不一定相等，*x*值也不一定有序。然而，负值间距和子区间被认为是负值积分。
`trapz(y)`	计算方法同上，但*x*值间隔为1。
`trapz(x,A)`	将**A**中每列的值带入**x**的函数算出其积分，并返回一组包含积分结果的向量。A的列向量必须和向量**x**的长度相同。
`Z=trapz(x,A,dim)`	在矩阵**A**中*dim*指定的维内进行数据积分。如果给定向量x，则x的长度必须与size(A, dim)相同。
`cumtrapz(A,dim)`	返回大小和**A**相同的数组，包含的是将矩阵**A**进行梯形积分的累积值。如果*dim*已给定，则在*dim*维内进行计算。
`quad(fcn,a,b)`	返回在区间[a, b]上g的积分近似值。字符串**fcn**包含一个与**g**相对应的MATLAB函数名，也就是预定义函数或者是M文件。这个函数接收一个向量参数，并返回一个向量结果。MATLAB利用辛普森规则执行递归的积分，计算误差为10^{-3}。
`quad(fcn,a,b,tol)`	求**g**的积分近似值，其相对误差由参数*tol*定义。否则，计算过程同上。
`quad(fcn,a b,tol,pic)`	求**g**的积分近似值，其相对误差由参数*tol*所定义。如果参数*pic*是非零值，则在图形中显示求值的点。
`quad(···,trace)`	如果*trace*是非零值，则画出积分图形。

quad8(···)	可以与quad一样用于相同的参数组合并返回相同的结果，但使用更高精度的方法。因此，如果被积函数的导数在某一区间内是不定的，例如：$q=\int_0^1 \sqrt{\sin x}dx$，使用此命令将会更好一些。quad和quad8 都要求被积函数在整个区间里是有限的。
dblquad(f, min1, max1,min2,max2, tol,trace,order)	计算双变量函数**f**的二重积分。函数中的第一个自变量用于内层积分。内层积分在*min*1 和*max*1之间进行，外层积分在*min*2和*max*2之间进行。变量*tol*指定相对误差。*trace*的使用方法与quad相同。根据字符串 **order**，对于相同的访问，dblquad能选择使用quad、quad8和许多用户定义的积分方法，并返回与quad相同的变量。

输入quaddemo可以看到一个演示实例。

■ **例**11.1

下面用不同的方法来计算下列积分：

$$\int_0^1 e^{-x^2}dx$$

(a) 使用trapz命令。首先创建一个有*x*值的向量。用5和10两个值进行计算：

```
x5 = linspace(0,1,5); x10 = linspace(0,1,10);
```

然后创建x的函数y:

```
y5 = exp(-x5.^2); y10 = exp(-x10.^2);
```

现在计算出积分值：

```
format long;
integral5  = trapz(x5,y5), ...
integral10 = trapz(x10,y10)
```

返回

```
integral5 =
   0.74298409780038

integral10 =
   0.74606686791267
```

(b) 使用quad命令。首先在M文件中创建函数。此文件integrand.m包含函数，如下：

```
function y = integrand(x)

y = exp(-x.^2);
```

首先以标准误差计算积分，然后再以指定误差计算积分。

```
format long;
integralStd = quad('integrand',0,1)
integralTol = quad('integrand',0,1,0.00001)
```

给出

```
integralStd =
   0.74682612052747

integralTol =
   0.74682414517798
```

(c) 使用quad8命令：使用在(b)中创建的**M文件**,然后输入：

```
integral8Std = quad8('integrand',0,1)

integral8Std =
   0.74682413281243
```

这是MATLAB所能给出的最精确的结果。

(d) 使用cumtrapz命令能很容易地计算出不同区间的积分。

```
x = 0:5;
cumtrapz(x)

ans =
        0    0.5000    2.0000    4.5000    8.0000   12.5000
```

(e) 计算二重积分：

$$\int_0^1 \int_0^1 e^{-x^2-y^2}\,dy\,dx$$

如图11-1。首先创建一个包含函数M文件:**integrand2.m**:

```
function y = integrand2(x,y)

y = exp(-x.^2-y.^2);
```

然后用quad命令计算对于固定的x值在y方向的一些积分值：

```
x = linspace(0,1,15);

for i = 1:15
  integral(i) = quad('integrand2',0,1,[],[],x(i));
end
```

现在已计算出在y方向的15个积分值。trapz命令能使用这些值来计算二重积分：

```
format short;
dIntegral = trapz(x,integral)


dIntegral =
    0.5575
```

输入下列语句可以得到一个积分区域的图形：

```
[X,Y] = meshgrid(0:.1:1,0:.1:1);
Z     = integrand2(X,Y);

mesh(X,Y,Z); view(30,30);
```

结果如图11-1所示。命令mesh和view定义在13.5节中。

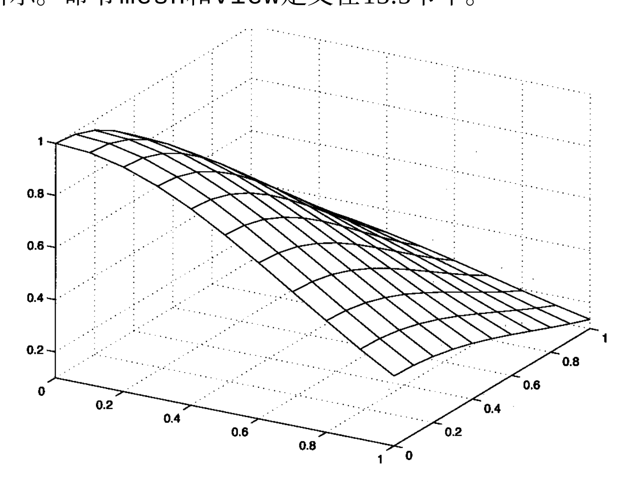


图11-1 函数e^{-x2-y2}在区间[0，1]×[0，1]的上的图形

■

不定积分 $\int_a^x f(t)dt$ 不能使用上面的命令来计算。 MATLAB中的数学符号工具箱和MATLAB的编辑器能提供处理这些积分的命令。

11.2 常微分方程组

下面来研究常微分方程系统**ODE**，该系统处理的是初始值已知的一阶微分方程。在本节中主要讨论这种类型的微分方程，同时也会举出两个**有关边界值问题的**例子。可以利用**ODE**系统创建稀疏线性系统方程来求解这些例子。

在数学符号工具箱中，有一些命令能给出常微分方程的符号解，即解以数学表达式的形式给出。

在下面的**初始值问题**中，有两个未知函数：$x1(t)$和$x2(t)$，并用以下式子表达其微分形式：

$$\frac{dx_i}{dt} = x_i'$$

在许多应用中，独立变量参数t表示时间。

$$\begin{cases} x_1' = f_1(x_1, x_2, t) \\ x_2' = f_2(x_1, x_2, t) \\ x_1(t_0) = x_{1,0} \\ x_2(t_0) = x_{2,0} \end{cases}$$

高阶的**ODE**能表达成第1阶的**ODE**系统。例如，有以下微分方程：

$$\begin{cases} x'' = f(x, x', t) \\ x(t_0) = x_0 \\ x'(t_0) = xp_0 \end{cases}$$

用x_2替换x'用x_1替换 x，就能得到：

$$\begin{cases} x_1' = x_2 \\ x_2' = f(x_1, x_2, t) \\ x_1(t_0) = x_0 \\ x_2(t_0) = xp_0 \end{cases}$$

这是一个第1阶的**ODE**系统。

对于某一时间间隔$0 \leq t \leq T$，初始值问题的解决方法是将时间分成一组有限和离散的时间点，例如用相同的时间间隔Δ*t*进行等分：

$$t_i = i\Delta t, \quad i = 0, \ldots, N$$

其中时间步长$\Delta t = T/N$，N为某一整数。这种导数能被**微分方程**的**可微分的商**所代替，微分方程表示在不同时间点的解。见例 11.2，给出更多的有关有限微分商的信息。这种方法的稳定性取决于Δ*t*的大小和所采用的数值方法，用这种方法能得到 **ODE**的近似值。

在许多应用中有一些**微分过程非常复杂的微分方程**，在某些区域里这些方程要求有非常小的时间步长Δ*t*。解决这些问题的困难在于问题中涉及不同的时间尺度，如解的导数可能有较大的变化。

MATLAB使用**龙格-库塔-芬尔格**(Runge-Kutta-Fehlberg)方法来解**ODE**问题。在有限点内计算求解，而这些点的间距由解本身来决定。当解比较平滑时，区间内使用的点数少一些；在解变化很快时，区间内应使用较多的点。

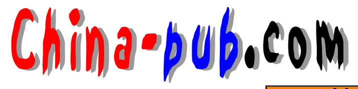
为了得到更多的有关何时使用哪种解法和算法的信息，推荐使用 `helpdesk`。所有求解方程通用的语法或句法在命令集 108中头两行给出。时间间隔将以向量 `t=[t0,tt]`给出。

命令`ode23`可以求解(2，3)阶的常微分方程组，函数 `ode45`使用(4，5)阶的龙格-库塔-芬尔格方法。注意，在这种情况下 **x**´是**x**的微分，不是**x**的转置。

在命令集108中`solver`将被诸如`ode45`函数所代替。

命令集108　龙格-库塔-芬尔格方法

命令	说明
`[time,X]=solver(str,t,x0)`	计算**ODE**或由字符串**str**给定的**ODE**的值。部分解已在向量**time**中给出。在向量time中给出部分解，包含的是时间值。还有部分解在矩阵 **X**中给出，**X**的列向量是每个方程在这些值下的解。对于标量问题，方程的解将在向量X中给出。这些解在时间区间 **t(1)**到**t(2)**上计算得到，其初始值是**x0**即**x(t(1))**。此方程组由**str**指定的M文件中函数表示出。这个函数需要两个参数：标量 *t*和向量**x**，应该返回向量**x**'(即x的导数)。因为对标量ODE来说，**x**和**x**'都是标量。在M文件中输入`odefile`可得到更多信息。同时可以用命令 `numjac`来计算雅可比函数。
`[t,X]=solver(str,t,x0,val)`	此方程的求解过程同上。结构 **val**包含用户给`solver`的命令。参见`odeset`和表11-1，可得更多信息。
`ode45`	此方法被推荐为首选方法。
`ode23`	这是一个比`ode45`低阶的方法。
`ode113`	用于更高阶或大的标量计算。
`ode23t`	用于解决难度适中的问题。
`ode23s`	用于解决难度较大的微分方程组。对于系统中存在常量矩阵的情况也有用。



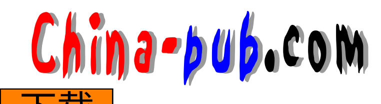
ode15s	与ode23s相同，但要求的精度更高。
ode23tb	用于解决难度较大的问题。对于系统中存在常量矩阵的情况也有用。
set=odeset(set1,val1,set2,val2,···)	返回结构**set**,其中包含用于**ODE**求解方程的设置参数。有关可用设置的信息参见表11-1。
odeget(set,'set1')	返回结构**set**中设置set1的值

有许多设置对odeset控制的**ODE**解是有用的，参见表11-1。例如，如果在求解过程中要画出解的图形，可以输入：inst=odeset('OutputFcn','odeplot');。

表11-1　ODE求解方程的设置参数

RelTol	给出求解方程允许的相对误差
AbsTol	给出求解方程允许的绝对误差
Refine	给出与输出点数相乘的因子
OutputFcn	这是一个带有输出函数名的字符串，该字符串将在求解函数执行的每步被调用：odephas2(画出2D的平面相位图),odephas3(画出3D的平面相位图),odeplot(画出解的图形),odeprint(显示中间结果)
OutputSel	是一个整型向量，指出哪些元素应被传递给函数，特别是传递给OutputFcn
Stats	如果参数Stats为on，则将统计并显示出计算过程中资源消耗情况
Jacobian	如果编写**ODE**文件代码以便F(t,y,'jacobian')返回*dF*/*dy*，则将Jacobian设置为on
Jconstant	如果雅可比数*df*/*dy*是常量，则将此参数设置为on
JPattern	如果编写**ODE**文件的编码以便函数F([],[],'jpattern') 返回带有零的稀疏矩阵并输出非零元素*dF*/*dy*，则需将Jpattern设置成on
Vectorized	如果编写 **ODE**文件编码以便函数 F(t,[y1,y2'···])返回[F(t,y1) F(t,y2·)]，则将此参数设置成on
Events	如果**ODE**文件中有带有参数'events'，则将此参数设置成on
Mass	如果编写**ODE**文件编码以实现函数F(t,[],'mass')返回*M*和*M*(*t*)，应将此参数设置成on
MassConstant	如果矩阵*M*(*t*)是常量，则将此参数设置为on
MaxStep	此参数是限定算法能使用的区间长度上限的标量
InitialStep	给出初始步长的标量。如果给定的区间太大，算法就使用一个较小的步长
MaxOrder	此参数只能被ode15s使用，它主要是指定ode15s的最高阶数，并且此参数应是从1到5的整数
BDF	此参数只能被ode15s使用。如果使用倒推微分公式而不是使用通常所使用的微分公式，则要将它设置为on
NormControl	如果算法根据norm(e)<=max(RelTol*norm(y),AbsTol)来步积分过程中的错误，则要将它设置为on

也可试用命令odedemo。



■ **例**11.2

(a) 求解下面的**ODE**：

$$\begin{cases} x' = -x^2 \\ x(0) = 1 \end{cases}$$

创建函数*xprim*1，将此函数保存在M文件**xprim1.m**中：

```
function xprim = xprim1(t,x)

xprim = -x.^2;
```

然后，调用MATLAB的**ODE**算法求解方程，最后画出解的图形：

```
[t,x] = ode45('xprim1',[0 1],1);

plot(t,x,'-',t,x,'o');
xlabel('time t0 = 0, tt = 1');
ylabel('x values x(0) = 1');
```

得到图11-2。MATLAB计算出的解用圆圈标记。在13.1节中介绍绘图命令`plot`。

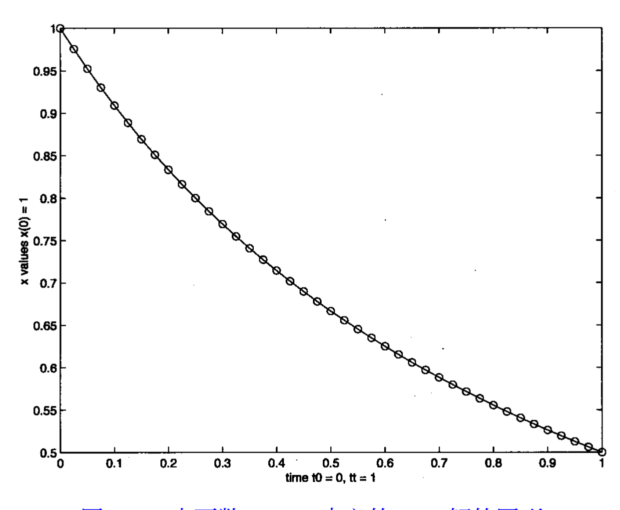


图11-2 由函数xprim1定义的ODE解的图形

(b) 解下面的**ODE**过程是等价的：

$$\begin{cases} x' = x^2 \\ x(0) = 1 \end{cases}$$

首先创建函数*xprim*2，并将其保存在M文件**xprim2.m**中：

```
function xprim = xprim2(t,x)

xprim = x.^2;
```

然后调用ODE的求解方程并画出其解的图形：

```
[t,x] = ode45('xprim2',[0 0.95],1);

plot(t,x,'o',t,x,'-');
xlabel('time t0=0, tt=0.95');
ylabel('x values x(0)=1');
```

得到图11-3。

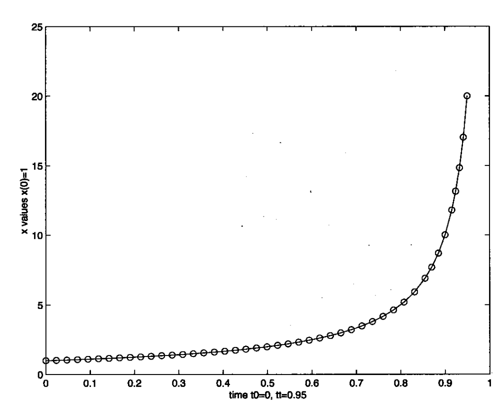


图11-3 由函数xprim2定义的ODE解的图形

注意，在MATLAB中计算出的点在微分绝对值大的区域内更密集些。

(c) 求解：

$$\begin{cases} x' = x^2 \\ x(0) = -1 \end{cases}$$

可使用与(b)中相同的函数。只要改变一下初始数据即可：

```
[t,x] = ode45('xprim2',[0 1],-1);

plot(t,x);
xlabel('time t0 = 0, tt = 1');
ylabel('x values x(0) = -1');
```

给出图11-4。

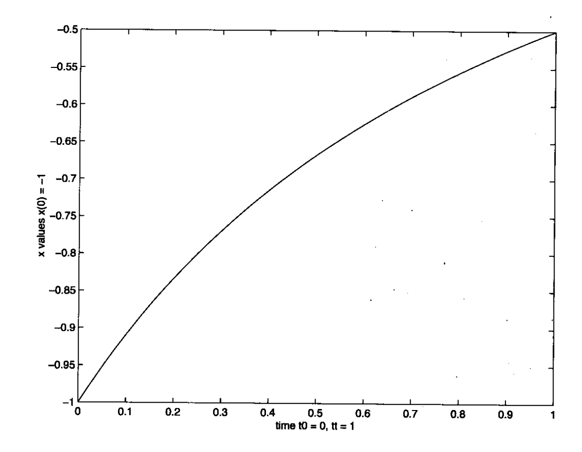
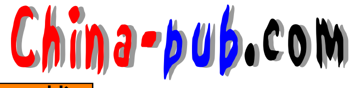


图11-4 给定新的初始数据，由函数xprim2定义的ODE解的图形



(d) 求解下面方程组并不很难：

$$\begin{cases} x_1' = x_1 - 0.1x_1x_2 + 0.01t \\ x_2' = -x_2 + 0.02x_1x_2 + 0.04t \\ x_1(0) = 30 \\ x_2(0) = 20 \end{cases}$$

这个方程组应用在人口动力学中，可以认为是单一化的捕食者——被捕食者模式。例如，狐狸和兔子。x_1表示被捕食者，x_2表示捕食者。如果被捕食者有无限的食物，并且不会出现捕食者。于是有$x_1'=x_1$，这个式子是以指数形式增长的。大量的被捕食者将会使捕食者的数量增长；同样，越来越少的捕食者会使被捕食者的数量增长。而且，人口数量也会增长。洛特卡和伏尔泰拉在20世纪20年代已对这些非线性的微分方程进行了研究。

创建函数*xprim*3，并将其保存在**M**文件**xprim3.m**中：

```
function xprim = xprim3(t,x)

xprim    = [ x(1) - 0.1*x(1)*x(2)  + 0.01*t; ...
            -x(2) + 0.02*x(1)*x(2) + 0.04*t];
```

然后调用一个ODE算法和画出解的图形：

```
[t,x] = ode45('xprim3',[0 20],[30; 20]);

plot(t,x);
xlabel('time t0=0, tt=20');
ylabel('x values x1(0)=30, x2(0)=20');
```

所得结果如图11-5所示。

在MATLAB中，也可以根据x_2函数绘制出x_1的图形。命令`plot(x(:,2),x(:,1))`可绘制出平面相位图，如图11-6所示。

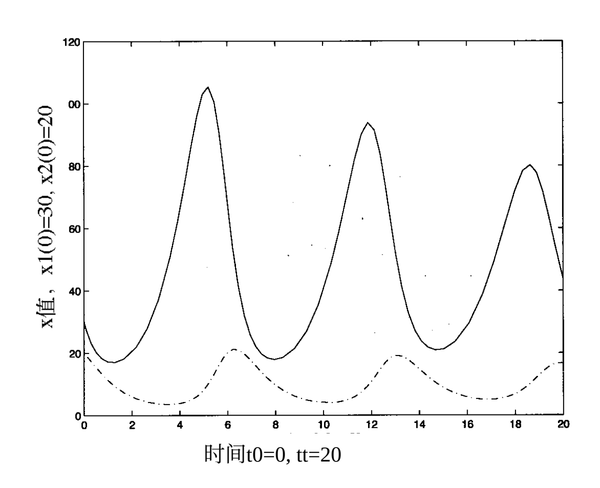


图11-5　函数**xprim3**定义的**ODE**解的图形

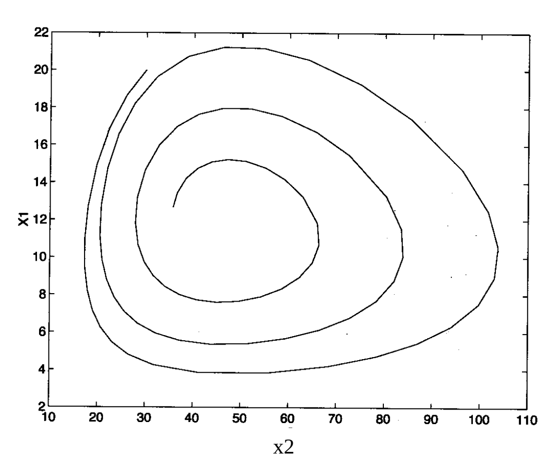


图11-6　由函数**xprim3**定义并根据函数x_2计算出的x_1值的曲线图

■

■ 例11.3

对于某些a和b值，下面的问题比较难解：

$$\begin{cases} x_1' = a - (b+1)x_1 + x_1^2 x_2 \\ x_2' = bx_1 - x_1^2 x_2 \\ x_1^0 = 1 \\ x_2^0 = 3 \end{cases}$$

方程由下面的**M**文件**stiff1.m**定义：

```
function stiff=stiff1(t, x)
global a;          % 变量不能放入参数表中
global b;
stiff=[0;0];       % Stiff必须是一个冒号向量
stiff(1) = a - (b+1)*x(1) + x(1)^2*x(2);
stiff(2) = b*x(1) - x(1)^2*x(2);
```

下面的**M**文件给出一个比较困难的问题：

```
global a; a = 100;
global b; b = 1;
tic;
[t,X] = ode23('stiff1',[0 10],[1 3]);
toc
size(t)
```

运行后得到的结果如下：

```
elapsed_time =
   72.1647

ans =
        34009           1
```

使用专门解决复杂问题的解法ode23s，将会得到较好的结果：

```
elapsed_time =
    1.0098

ans =
   103      1
```

■

对于**边界值问题**，除了微分方程，还有在边界处的值。在一维下这意味至少有两个条件。现在举两个如下的例子：

• 假设要研究一根杆的温度分布情况。这根杆一端的温度是 T_0，另一端的温度是 T_1；如图11-7所示。

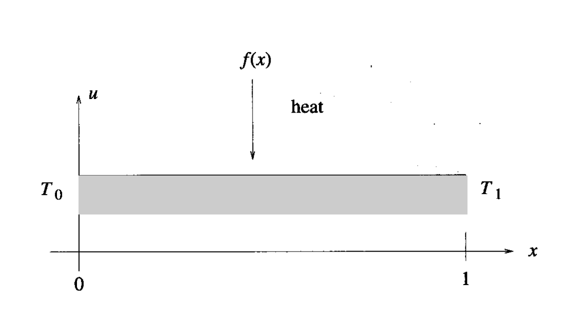
令$y(x)$表示这根杆的温度，函数$f(x)$表示加热源。

从时间t=0开始，在相当长的时间内加热这根杆，直至达到平衡状态。这就是所谓的定常值或稳定状态。这个定常值可由下面的方程模型表示：

$$\begin{cases} -y''(x) = f(x), \quad 0 < x < 1 \\ y(0) = T_0 \\ y(1) = T_1 \end{cases}$$

假设这根杆两端为：x=0和 x=1。

• 假设在其两端有一根固定的柱子(或者可以看成是一个连接两个岛屿的桥)，如图11-8所示。



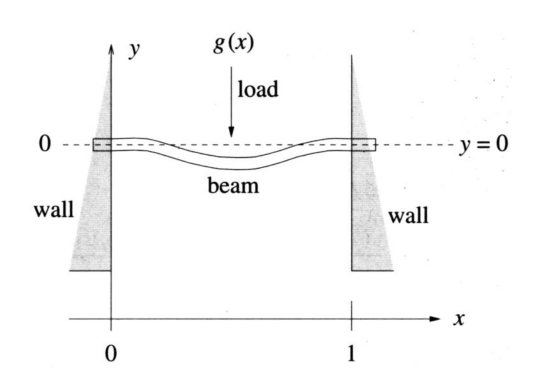


图11-7 在一根杆上的温度分布图

图11-8 在柱子上加载重量(相当于桥上交通堵塞)

令$y(x)$表示加载函数$g(x)$后弯曲的柱子。此问题需要有两个关于此柱子两端的边界条件。假设这根柱子非常牢固地固定在墙上，即 y在墙上的导数是0。可以得到下面的**ODE**，其中介绍了自然协调系统：

$$\begin{cases} y''''(x) = g(x), \quad 0 < x < 1 \\ y(0) = y'(0) = y(1) = y'(1) = 0 \end{cases}$$

由于存在边界值问题，不可能象解决初始值问题一样一次只执行一步地来解决问题。因此必须解一个同时给出所有未知参数的方程组。

假设有一个**ODE**，函数$y(x)$是它的解。用**近似的差分**来代替微分方程就能解这个**ODE**问题。为了能这样做，必须将区间分成有限数量的点：x_0，$x_1,\cdots,x_M$，其中$x_{j+1}=x_j+\Delta x$,然后计算出区间内各点的近似值$y_j \approx y(x_j)$，并给出确定的边界值，如y_0和y_M或更多的值；如图11-9所示。

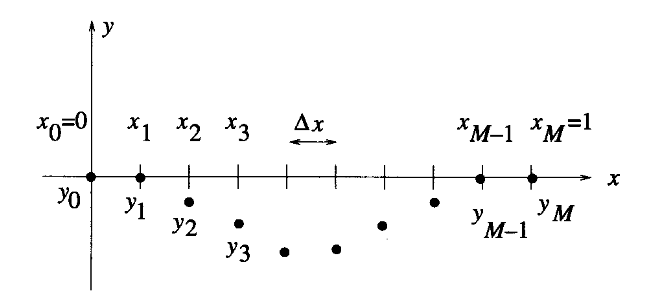


图11-9 将计算区间[0，1]分成M等份

解$y(x)$的导数可由**有限的差分**代替，如下：

$$\begin{cases} y'(x_j) \approx \dfrac{y(x_{j+1}) - y(x_j)}{\Delta x} \\ y''(x_j) \approx \dfrac{y(x_{j+1}) - 2y(x_j) + y(x_{j-1})}{\Delta x^2} \\ y''''(x_j) \approx \dfrac{y(x_{j+2}) - 4y(x_{j+1}) + 6y(x_j) - 4y(x_{j-1}) + y(x_{j-2})}{\Delta x^4} \end{cases}$$

如果用这些差分方程来代替**ODE**中的导数，就能得到一个所有未知y_j的方程组。其系数矩阵是一个有序区间，此区间的宽度决定于这个微分方程的导数个数。

■ **例**11.4

根据前面的温度模型的方程研究一下杆的温度分布，将所有的导数换成不同的差分并得到：

$$\begin{cases} -\dfrac{y_{j+1}-2y_j+y_{j-1}}{\Delta x^2}=f_j, & j=1,\ldots,M \\ y_0=T_0 \\ y_M=T_1 \end{cases}$$

其中$f_j=f(x_j)$。为了简单起见，设M=6，即给定y_0和y_6，而y_1，y_2，… ,y_5为未知变量。于是就有：

$$\begin{cases} -y_0 + 2y_1 - y_2 = \Delta x^2 f_1 \\ -y_1 + 2y_2 - y_3 = \Delta x^2 f_2 \\ -y_2 + 2y_3 - y_4 = \Delta x^2 f_3 \\ -y_3 + 2y_4 - y_5 = \Delta x^2 f_4 \\ -y_4 + 2y_5 - y_6 = \Delta x^2 f_5 \end{cases}$$

注意，$y_0=T_0$和$y_M=T_1$必须移到方程组的右边。此时得到的矩阵是一个对角矩阵，其对角线上的元素为2，并且上一对角线和下一对角线上的元素为 1。

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} \Delta x^2 f_1 + T_0 \\ \Delta x^2 f_2 \\ \Delta x^2 f_3 \\ \Delta x^2 f_4 \\ \Delta x^2 f_5 + T_1 \end{pmatrix}$$

下面解此问题的文件**temperature.m**。用户必须先给出分段数及$f(x)$(用点符号)，最后给出T_0和T_1。有关稀疏矩阵更多信息参见第 9章。

```
% 杆上的温度分布，用T0和T1分别表示两端温度
% 这根杆放在x坐标的0和1区间上，并被分成M个子区间，每一个子区间的长度为1/M
% 创建稀疏矩阵方程Ax=b并求解
% 矩阵A是对角阵，并以稀疏矩阵的形式存储

clear;

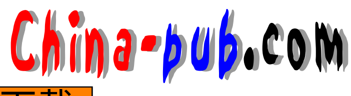
M = input('Give the number of subintervals (M): ');
deltax = 1/M;
xx = 0:deltax:1;

funcStr = input('Give f(x),
          the extra heat source (e.g., x.^3): ', 's' );

T0 = input('Give y(0) (left):  ');
T1 = input('Give y(1) (right): ');

% 构造对角矩阵A和方程右边b

vectorOnes = ones(M-1,1);
A = spdiags([-vectorOnes, 2*vectorOnes, -vectorOnes],
            [-1 0 1],M-1,M-1);
```




```
x=xx(2:end-1);          % x为区域内的值。
f=eval(funcStr);        % 相应的f(x)值。
b=deltax^2f;
b(1)=b(1)+T0;           % 对边界值x=0,x=1进行特殊处理。
b(end) = b(end) + T1;
b = b';

% 解线性方程
y=A\b;                  % y在区间内：j=1,2,···,M-1。
y=[T0;y;T1];            % y在整个区间内：0<=x<=1。
clf;
% 上面图形表示外部热源。
% 下面图形表示杆上的热分布。

subplot(2,1,1);
plot(x,f);
grid on;
title('External heat source f(x).', 'FontSize', 14 );

subplot(2,1,2);
plot(xx,y,'r');
grid on;
title('Temperature distribution in a rod.', 'FontSize', 14);
```

将区间分成100等份，根据方程$f(x)=x^2+\sin(10\pi x)$在图11-10中可以得到解。

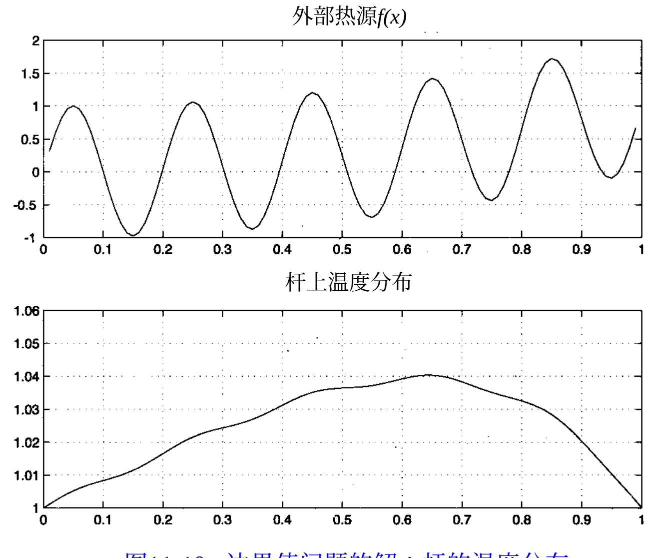


图11-10　边界值问题的解：杆的温度分布

■ **例**11.5

如果把前面柱子例题中的导数替换掉，即用 y_j近似值表示解，就可以得到：

$$\begin{cases} \dfrac{y_{j+2}-4y_{j+1}+6y_j-4y_{j-1}+y_{j-2}}{\Delta x^4}=g_j, & j=2,\ldots,M-2 \\ y_0=\dfrac{y_1-y_0}{\Delta x}=y_M=\dfrac{y_M-y_{M-1}}{\Delta x}=0, \end{cases}$$

将其重写为：

$$\begin{cases} y_{j+2}-4y_{j+1}+6y_j-4y_{j-1}+y_{j-2}=\Delta x^4 g_j, & j=2,\ldots,M-2 \\ y_0=y_1=y_{M-1}=y_M=0 \end{cases}$$

这是一个真正的线性方程组，其中用 M-3个方程来解M-3个未知数：y_2，$y_3,\cdots,y_{M-2}$。如果M=10 ，就有：
解是一个5对角矩阵，使用\运算符能很快且有效地解出此方程。

$$\begin{pmatrix} 6 & -4 & 1 & 0 & 0 & 0 & 0 \\ -4 & 6 & -4 & 1 & 0 & 0 & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & 0 & 0 & 1 & -4 & 6 & -4 \\ 0 & 0 & 0 & 0 & 1 & -4 & 6 \end{pmatrix} \begin{pmatrix} y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{pmatrix} = \Delta x^4 \begin{pmatrix} g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \\ g_8 \end{pmatrix}$$



第12章　MATLAB程序设计

MATLAB有一些命令可以来控制 MATLAB语句的执行，如条件语句、循环语句和支持用户交互的命令，本章将介绍这些命令。 MATLAB是一种高级的程序设计语言，能帮助用户解决矩阵问题或其他问题。那些熟悉其他编程语言的用户，如熟悉 Pascal、C++、FORTRAN等，对理解本章内容有一定的优势。但是确信这部分内容能够让所有的读者理解和掌握。

12.1　条件控制语句

MATLAB中由if语句做出判断。If语句的基本格式如下：

```
if logical expression
   statements
end
```

注意，在if和*logical expression*(逻辑表达式)之间要有一个空格。 *statement*(程序语句)可以是一个命令，也可以是由逗号、分号隔开的若干命令或者是‘ returns ’。只有当逻辑表达式为**true(**真)时，才能执行这些命令。逻辑表达式可以是一个标量、一个向量或者一个矩阵。如果逻辑表达式的所有元素为非零值，它才为 **true(**。

if语句也可以写成一行。

```
if logical expression, statements, end
```

当然，通常前一种形式使得 MATLAB程序更加结构化和易读。

■ **例**12.1

假设定义 $m \times n$的矩阵**A**。下面的语句是判断矩阵 A的第1列元素是否为0，若全为0，则从矩阵**A**中删除第1列：

```
if A(:,1) == 0
  A = A(1:m,2:n)
end
```

或者写成一行：

```
if A(:,1) == 0, A = A(1:m,2:n), end
```

■

if语句可以与elseif或else组合起来用于更复杂的上下文语句中。可能有如下的结构存在：

```
if logical expression
   statements 1
else
   statements 2
end
```

如果逻辑表达式为**true,** 则执行*statements* 1中的命令语句；如果为**false**则执行*statements* 2中的语句。

考虑下面的if语句：

```
if logical expression 1
   statements 1
elseif logical expression 2
   statements 2
end
```

当*logical expression* 1为true时，执行*statements* 1中的命令；如果*logical expression* 1为false并且 *logical expression* 2为true时，执行*statements* 2。

注意，elseif必须写成一个单词，如果分开写成else if，将会被解释成不同的意思。命令elseif不像else if一样需要一个额外的end。

另外if语句可以被嵌套成下面的形式：

```
if logical expression 1
   statements 1
elseif logical expression 2
   statements 2
else
   statements 3
end
```

更复杂的情况如下：

```
if logical expression 1
   statements 1
   if logical expression 2
      statements 2
   else
      statements 3
   end
else
   statements 4
end
```

■ **例**12.2

(a) 如果A为非奇异矩阵，就能解方程 **Ax=b**；否则要取决于扩展矩阵(**A b**)的梯形形式行的个数。提示：如果一个矩阵是方阵或为满秩的，则它为非奇异矩阵。

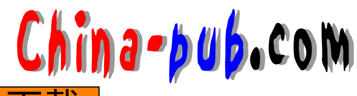
```
% 给出矩阵A和方程右边b。
s = size(A)

if (s(1) == s(2)) & (rank(A) == s(1))
  x = A\b
else
  rref([A b])
end
```

(b) 如果矩阵A的行列式为0，则计算特征值为0的个数：

```
if det(A) == 0
  length(find(eig(A) == 0))
end
```

■



另一种条件语句是switch-case语句，如下：

```
switch logical expression (scalar or string)
case value 1
   expression 1
case value 2
   expression 2
...
otherwise
   expression
end
```

*logical expression*经过计算给出一个标量或字符串作为结果。将这个结果与*value* 1, *value* 2, ...进行比较，如果它们匹配，则执行相应的case下的语句*expression*。如果没有匹配的，则执行otherwise下的语句。

如果*expression*的结果是一个标量，将通过检查：*expression*= =*value*来决定执行的表达式。如果表达式的结果是一个字符串，那么用 strcmp(*expression*, *value*)来检查。测试结果为真，则执行相应的表达式，而其他case语句中的表达式将不会被执行。

通过将不同的值放入细胞矩阵，就能用case语句与不同的值进行比较；见例12.3。

■ **例**12.3

检测掷一次骰子所得的点数是单数还是双数：

```
function dicetest(result)

switch result
case {1,3,5}
  disp('odd number of eyes')
case {2,4,6}
  disp('even number of eyes')
otherwise
  disp('What kind of dice do you have?')
```

运行这个函数可以得到如下结果：

```
dicetest(1)

odd number of eyes

dicetest(4)

even number of eyes

dicetest(7)

What kind of dice do you have?
```

■

如果表达式出错，可以使用try/catch组合，其形式如下：

```
try
   expression 1
catch
   expression 2
end
```

MATLAB开始执行*expression* 1，但如果有错误，错误信息将被存储在 lasterr中，并且执行*expression* 2。

12.2 循环语句

MATLAB有两个命令**for**和**while**能反复执行语句。在逻辑控制下，这些命令能灵活地一次或多次执行语句。

命令for与大多数的程序设计语言中的 do或for命令一样。这个命令就是反复执行一条语句或一组语句，而执行的次数已预先定义好。以 end结束这组语句。

for循环通常的语法为：

```
for variable = expression
    statements
end
```

象if语句一样， for语句也能写在一行上：

```
for variable = expression, statements, end
```

在for和*variable*之间需要有一个空格。这里的 *variable*是循环变量名。在表达式中给出循环的初始值、步长和终值。这个步长可为负数或单位值。如果为单位值，循环变量每次迭代将增加1。通常我们用冒号来定义 *expression*，例如i:j:k或i:j，参见4.3节 。

表达式中的列值被一个一个地存放在循环变量中。因此，可以用一个矩阵来代替表达式。例如下面的语句：

```
for v = A, ..., end
```

就等价于：

```
for j = 1:n, v = A(:,j); ..., end
```

当表达式用冒号来表示时，那么列值都是标量，例如MATLAB中的语句：for v=i:j: k。

循环是可以嵌套的：

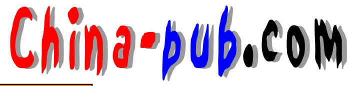
```
for variable I = expression A
    statements 1
    for variable II = expression B
        statements 2
    end
    statements 3
end
```

■ **例**12.4

(a) 下列矩阵有三个非零对角值(这是一个三对角阵):

$$\mathbf{A} = \begin{pmatrix} 5 & 1 & 0 & 0 & 0 \\ 1 & 5 & 1 & 0 & 0 \\ 0 & 1 & 5 & 1 & 0 \\ 0 & 0 & 1 & 5 & 1 \\ 0 & 0 & 0 & 1 & 5 \end{pmatrix}$$

这个矩阵可循环使用命令for来创建。这种方法在任何标准的程序设计语言中都是一样的。




```
A = [];

for k = 1:5
  for j = 1:5

    if k == j
      A(k,k) = 5;
    elseif abs(k-j) == 1
      A(k,j) = 1;
    else
      A(k,j) = 0;
    end

  end
end
```

这里的分号';是非常重要的。如果这些赋值语句没有分号，矩阵 **A**将在屏幕上输出25次，每一次**A**中的元素将被赋值一次。

同样也会遇见由于不注意使用 for循环而导致无效操作的例子。通过定时器时钟就能清楚地计算出花费的时间。例如， for循环，见例12.21。下面的命令能完成上面同样的事情，并且更加有效：

```
A = zeros(5);

for k = 1:4
  A(k,k) = 5;
  A(k,k+1) = 1;
  A(k+1,k) = 1;
end

A(5,5) = 5;
```

这个矩阵能通过更加快速有效的方法得到，但是使用命令 diag更加清楚。

```
A = [];

A = diag(5*ones(5,1)) + diag(ones(4,1),1) + ...
    diag(ones(4,1),-1);
```

这种结构的大矩阵应该创建成稀疏矩阵；参见第 9章。

(b) 在区间[−2,−0.75]内，步长为0.25，对函数$y=f(x)=1+1/x$求值，并列表。将所得x值和y值分别存入向量r和s中，并列表显示：

```
r = []; s = [];

for x = -2.0:0.25:-0.75
  y = 1 + 1/x;
  r = [r x];
  s = [s y];
end

[r; s]'
```

此表也能够不用for循环语句创建，其结果为：

```
ans =
   -2.0000    0.5000
   -1.7500    0.4286
   -1.5000    0.3333
   -1.2500    0.2000
   -1.0000         0
   -0.7500   -0.3333
```

(c) MATLAB 命令sum(A)给出一行向量，向量的元素是矩阵 **A**每一列元素的和。用下面的程序能得到相近的结果：

```
A = [1 2 3;4 5 6]

sum_v = [];

for v = A
  sum_v = [sum_v sum(v)]
end

disp('Compare w/ sum(A):');
disp(sum(A));
```

其结果是：

```
A =
     1     2     3
     4     5     6

sum_v =
     5

sum_v =
     5     7

sum_v =
     5     7     9

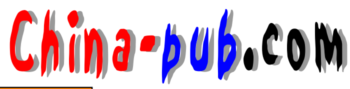
Compare w/ sum(A):
     5     7     9
```

(d) 将下列MATLAB命令保存在文件**qrmethod.m**中：

```
% 矩阵A及整数m和n应在调用该文件之前定义好。
% 在变换成上海森伯形式之后使用QR方法。
% n步以后结束。
% 每隔m步输出结果。

A = hess(A);

for i = 1:n
  [Q,R] = qr(A);
  A = R*Q;
  nd = norm(diag(A,-1));
```


下载


```
  if rem(i,m) == 0
    A, i, nd
  end

end
```

下面的程序是用来完成非移位的QR方法(见8.2节)30次迭代，每隔15次输出结果：

```
A0 = [-9 -3 -16;13 7 16;3 3 10 ];
m = 15; n = 30;
format long;
A = A0;
qrmethod;

A =
   9.98997467074377   22.62301237506363  -15.53274662438004
   0.00708686385759   -5.98568512552925    5.77401643542405
                  0    0.00741470005235    3.99571045478546

i =
    15

nd =
   0.01025677416162

A =
  10.00000471624660   22.62743993744967   15.51339551121122
  -0.00000333488655   -6.00001449452640   -5.77348898879412
                  0    0.00001693654612    4.00000977827978

i =
    30

nd =
     1.726175143943722e-05
```

(e) 在7.5节中定义了命令planerot。这个算法是使用该命令来返回一个矩阵，该矩阵的0元素都在任何作为参数输入的 $m \times n$ 矩阵的主对角线的下方。

```
function B = Givens(A)
%
% 如果将A矩阵乘函数B返回的矩阵，就能将大小为 m×n的A矩阵变成上三角阵。也就是说，根据 Q = B和´
R=B*A，其中满足Q*R=A，该函数可以对A进行QR分解。
[m,n] = size(A);
B     = eye(m);

for j = 1:n
  for i = j:m
    for k = (i+1):m
      G      = eye(m);
      Plan   = planerot([A(j,j) A(k,j)]');
                                              %找出2×2的矩阵
      G(j,j) = Plan(1,1); G(k,j) = Plan(2,1);
      G(j,k) = Plan(1,2); G(k,k) = Plan(2,2);
```




```
                                  % 在m×m矩阵中正确定位
      B = G*B;
      A = G*A ;        % <- To see the step-by-step reduction
                       % of AA remove this semicolon.
    end
  end
end
```

在这个算法中，每一步的循环将**A**中的两个元素取出构成2×2的矩阵，然后将矩阵主对角线下的一个元素赋值为零。这个结果矩阵可用来创建 QR因式分解。

定义一个检测矩阵**Atest**：

$$\mathbf{Atest} = \begin{pmatrix} 1 & 2 & 3 & 1 & 2 & 3 \\ 4 & 4 & 1 & 2 & 2 & 1 \\ 7 & 6 & 3 & 2 & 1 & 1 \\ 1 & 2 & 1 & 0 & 0 & 2 \end{pmatrix}$$

下面的命令为：

```
Atest = [1 2 3 1 2 3;4 4 1 2 2 1;
         7 6 3 2 1 1;1 2 1 0 0 2];

Giv = Givens(Atest);
Q   = Giv', R = Giv*Atest
QR = Q*R                          % 仅仅是检测
```

给出MATLAB输出：

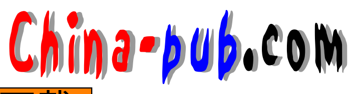
```
Q =
    0.1222    0.6630    0.6674    0.3162
    0.4887    0.1842   -0.5721    0.6325
    0.8552   -0.2947    0.2860   -0.3162
    0.1222    0.6630   -0.3814   -0.6325

R =
    8.1854    7.5745    3.5429    2.8099    2.0769    1.9547
    0.0000    1.6208    1.9523    0.4420    1.3997    3.2047
   -0.0000   -0.0000    1.9069    0.0953    0.4767    0.9535
    0.0000    0.0000    0.0000    0.9487    1.5811    0.0000

QR =
    1.0000    2.0000    3.0000    1.0000    2.0000    3.0000
    4.0000    4.0000    1.0000    2.0000    2.0000    1.0000
    7.0000    6.0000    3.0000    2.0000    1.0000    1.0000
    1.0000    2.0000    1.0000         0         0    2.0000
```

注意，通过乘Q和R我们又能得到初始矩阵**Atest**，因此**QR**=**Atest**。

(f) 以下程序通过使用两个for循环和平面组合来画出雪花图形。这个算法生成 Helge von Koch曲线，这是一个不规则碎片例子。程序中使用的图形命令定义在第 13章中，这里的注释主要说明其功能。该算法将当前几何图形每一面分成了相同的三个部分。第一部分和最后部



分是新几何学的两个方面。中间的部分用等边的三角形的两个边替代，如图 12-1所示。

如果将迭代进行下去，几乎平面的每一部分都将被覆盖到。事实上，不规则碎片的尺寸为1.2619，比1大一点而比2小。

```
% 文件 : Koch.m
% 该程序画出Helge von Koch 雪花，一个不规则碎片图形

clear;                         %删除旧变量

%向量在平面中新定义一个三角形。这是开始的几何状态。

new = [0.5+(sqrt(3)/2)*i,-0.5+(sqrt(3)/2)*i,...
       0,0.5+(sqrt(3)/2)*i];

plot(new);                     %画出三角形并等待0.5秒
pause(0.5);

% 迭代5次：                      向量old是前一次迭代

for k = 1:5;
  old = new;
  [m,n] = size(old);
  n = n - 1;

  % old定义了图中的n-1条边。
  % 对每条边：定义4个新点(其中一个是'old')。

  for j = 0:n-1;
    diff = (old(j+2)-old(j+1))/3;
    new(4*j+1) = old(j+1);
    new(4*j+2) = old(j+1) + diff;
    new(4*j+3) = new(4*j+2) + diff*((1-sqrt(3)*i)/2);
    new(4*j+4) = old(j+1) + 2*diff;
  end;

  % 向量new的最后一个元素与向量old的最后一个元素相同。

  new(4*n+1) = old(n+1);

  plot(new);                   %画出新图并等待0.5秒。
  pause(0.5);
end;

% 移开坐标轴，并使其等长度，图形会更匀称。

axis off; axis square;
```

执行程序，就可得到一个逐渐复杂的图形。

图12-2给出了最后的图形。

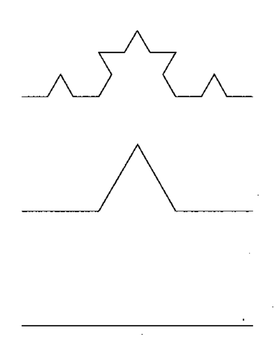


图12-1 von Koch算法对直线的两次迭代的结果图

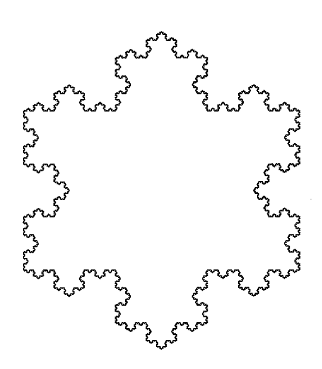


图12-2 5次迭代后的Helge von Koch不规则碎片图形，其原始几何图形为三角形

■

只要逻辑表达式为真，while命令将反复执行程序语句。像for语句一样程序体由一个end来结束。使用while循环来表示整个while语句，具体形式如下：

```
while, statements, end
```

通常，while循环有如下形式：

```
while logical expression
        statements
end
```

将其写在一行的形式为：

```
while logical expression, statement, end
```

while循环能够像for循环一样嵌套：

```
while logical expression A
        statements 1
        while logical expression B
                statements 2
        end
        statements 3
end
```

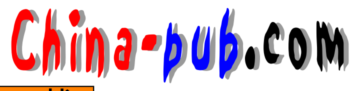
■ **例**12.5

(a) 构造一个特征值在1和－1之间的2×2的随机矩阵，可以用下面的迭代来实现：

```
A=rand(2);          % 构造一个特征值在1和-1之间的矩阵。
while max(abs(eig(A))) >= 1
   A = rand(2);
end

e = eig(A);
TheText = ['lambda_1 = ',num2str(e(1)), ...
          ', lambda_2 = ',num2str(e(2))];

A                   % 输出显示矩阵及其特征值。
```




```
disp(TheText)
```

运行程序，得到：

```
A =
    0.1517    0.6628
    0.2098    0.5295

lambda_1 = -0.077395, lambda_2 = 0.7586
```

其中变量lambda_1 和lambda_2是矩阵**A**的特征值。

加入一个变量用来计算迭代的次数。注意：如果程序再运行一次，会得到不同的结果。

```
A = rand(2); niter = 1;

while max(abs(eig(A))) >= 1
  disp(['Step ' num2str(niter)]);
  disp(['Eigenvalues: ' num2str(eig(A)', 5)]);
  A = rand(2);
  niter = niter + 1;
end

disp(['Final result - step ' num2str(niter)]);
disp(['Eigenvalues: ' num2str(eig(A)', 5)]);
```

结果是：

```
Step 1
Eigenvalues: 1.3871    -0.41031
Step 2
Eigenvalues: -0.18924      1.2166
Step 3
Eigenvalues: 1.0151    -0.11415
Final result - step 4
Eigenvalues: 0.054835     0.95675
```

(b) 函数ln(1+x)的Maclaurin序列为：

$$\ln(1+x)=\sum_{k=1}^{\infty}\frac{(-1)^{k+1}x^k}{k}$$

用x=0.5带入，并把Maclaurin序列的各项相加，直至要加的下一项系数小于内建变量 *eps*。计算出所加项的个数。用下面的方法来实现：

```
lnsum = 0; x = 0.5; k = 1;

while abs((x^k)/k) >= eps
  lnsum = lnsum + ((-1)^(k+1))*((x^k)/k);
  k = k + 1;
end

disp(['The sum = ',num2str(lnsum),...
      ', number of iter = ',num2str(k) ]);
```

给出的结果为：




```
The sum = 0.40547, number of iter = 47
```

检验这个结果：

```
ln = log(1.5)

ln =
    0.4055
```

■

有时在循环正常结束前终止循环是有用的，这可以用命令 break来实现。如果break命令用于嵌套循环的内部循环，那么只能终止内部循环，外部循环仍然继续。

应该尽量避免使用break，因为使用命令break的程序通常不易理解和维护。这样的程序通常被改写成没有break的程序。

■ **例**12.6

通过迭代求机器最小正数。

(a) 使用break的for循环：

```
macheps = 1;

for i = 1:1000
  macheps = macheps/2;

  if macheps + 1 <= 1
    break
  end

end

macheps = macheps*2
```

在Sun SPARC工作站上运行的结果为：

```
macheps =
   2.2204e-16
```

(b) 未使用break的while循环：

```
macheps = 1;

while macheps + 1 > 1
  macheps = macheps/2;
end

macheps = macheps*2
```

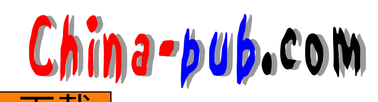
■

12.3 M文件的其他相关内容

在2.9节中介绍了M文件。在本节中将涉及到与M文件相关的其他方面的内容。

inline命令可以不用M文件就能创建函数；参见5.1.4节。

MATLAB能处理递归函数。这样的函数能调用本身，但要通过改变一些判断条件以防止该程序进入死循环。



■ **例**12.7

将下面的M文件命名为sqpulse.m：

```
function f = sqpulse(n,x)

% 递归函数用于求和：
% 1/2 + 2/pi cos(x pi) + ... + 2sin(n pi/2)cos(n x pi).
% For n --> inf 这将等于阶跃的平方

if (n == 1)
  f = 1/2 + 2/pi*cos(x*pi);           % 递归终止准则
else
  f = 2*sin(n*pi/2)/n/pi*cos(n*x*pi) + ...
      sqpulse(n-1,x);
end
```

如果n足够大，对x [−0.5,0.5]，函数将返回1。对于通过加上一个偶数也可变成属于这个集合的其他的x值，也就是如果x=−1.75， 函数将给出`sqpulse(n,x)=1`，因为−1.75+2等于0.25。对于其他所有的数值，函数**sqpulse**都是0；如图12-3所示。

如果n选得太小，阶跃的平方将会是正弦的，因为阶跃的平方是由余弦函数得到的。

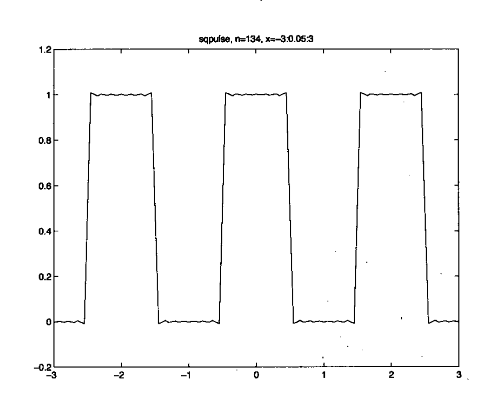


图12-3 用标题中的x和n值，函数plot(x,sqpulse(n,x)) 的图形

■

已经在M文件中加入很多注释。所有注释是以百分号 %开头的：

% 注释。

在编写程序时加入注释是很好的习惯，这有助于以后对程序功能和程序运行的理解。

`lookfor`命令(见2.7节)能浏览所有规定的M文件中第一行注释行。因此，将关键词放入第一行注释行是比较好的。

■ **例**12.8

```
% Course:      Applied Linear Algebra; Uppsala Univ.
% Assignment: homework #7 - LU-decomposition
% Date:        980505
% Author:      Tomas P.
% File name:   assignment7.m
```




```
%------------------------------------------------------------
% diary assignment          % Stores output in file "assignment".

txt1 = ...
    sprintf('\nAssignment #7, Syst. of Eq., T.P. 980505.\n');
txt2 = ...
    sprintf('Ax=b <==> (LU)x=b <==> (i) Ly=b + (ii) Ux=y.\n');

disp([txt1 txt2]);

A = [4 3;1 2];              % 创建系统矩阵A
b = [5;10];                 % 右边向量b

[L,U] = lu(A);              %矩阵A的LU分解

%L, U                        % 显示L和U

y = L\b;                        % (i) 前除
x = U\y;                        % (ii) 后置换
disp('Solution of Ax=b:');      % 写出结果向量
disp(x);

%x2 = A\b                       % 检查解

%程序结束
```

显示结果为：

```
Assignment #7, Syst. of Eq., T.P. 980505.
Ax=b <==> (LU)x=b <==> (i) Ly=b + (ii) Ux=y.
Solution of Ax=b:
    -4
     7
```

■

在调试过程中可使用命令符号‘ % ’。用关键字命令‘ commenting away ’能跟踪发现错误；参见12.7节。

当MATLAB第一次执行一个函数时，系统将创建一个该函数编译后的形式以便下一次调用该函数。可以将编译后的变量存入一个文件，也叫做 P文件。P文件的使用与M文件使用相同，但不能列出P文件。如果想要隐藏代码，这种方法是非常好的。

命令集109 **P文件**

`pcode fun1 fun2··`	编译函数**fun1**, **fun2**, ···,并以相同的名字**fun1**, **fun2**, ···将其存成文件，但后缀名为‘ .p ’。
`－inplace`	如果给出－ inplace，则象**M**文件一样将所有的 P文件保存到相同的路径下。
`[M,MEX]=inmem`	返回一个细胞矩阵 **M**，包含的是内存中编译过的 **M**文件的文件名字符串。如果给定 **MEX**，则返回已加载的**MEX**文件列表；参见15.2.1节和15.3.1节。

命令echo用来切换正在执行的命令的显示和不显示。当回显为打开状态时，所有的命令和注

释将被显示在屏幕上，这种方法对于调试程序非常有用。命令echo可写成echo on和echo off 。

函数文件不会被以上操作所影响。下面是函数文件的保留字：

命令集110 从函数文件中回显

`echo fname on`	打开函数**fname.m**的显示模式。
`echo fname off`	关闭函数**fname.m**的显示模式。
`echo fname`	切换函数**fname.m**开或关的显示模式。
`echo on all`	打开所有函数的显示模式。
`echo off all`	关闭所有函数的显示模式。

通常使用clear命令能将内存中的所有M文件清除。这种清除方法可以用下面的命令来控制：

命令集111 M锁定文件

`mlock`	锁住正在运行的M文件以便其不会被`clear`命令作用。
`mlock` *`filename`*	锁住M文件**filename**。
`munlock`	解锁正在运行的M文件，以便调用`clear`命令将其清除。
`munlock` *`filename`*	解锁M文件**filename**。
`mislocked`*`filename`*	如果正在运行的 M文件对于给出的 **filename**，或者**filename**，处于锁定状态，则都返回1；否则返回0。

一个函数可以有0个、1个或者多个参量(参数)，调用同一个函数可以带有不同个数的参量。例如，函数`triu(A)`返回一个上三角矩阵，而`triu(A,1)`则返回一个严格的上三角矩阵。

命令集112 参数个数

`nargin`	这是一个变量，指定调用函数所带参数的个数。
`nargout`	这是一个变量，指定调用函数所返回的参数的个数。
`inputname(x)`	返回输入表上数字x所在位置的输入参数变量的名字。如果用一个表达式代替已命名的参数，则返回一个空字符串。
`errorstr= nargchk(min, max,number)`	用来控制函数的输入参数的个数。参数 *number*是由nargin指定的输入参数的个数。如果 *number*的值超过*min*到*max*的区间范围，系统将返回一个错误字符串 **errorstr**，否则将返回一个空矩阵。
`varargin`	是函数可带有任意多个输入参数的细胞矩阵。
`varargout`	是函数可带有任意多个输出参数的细胞矩阵。

■ **例**12.9

可以将参数的可变个数定义成默认值，如果没有特别指定，就使用这个值。存放在**random.m**中的函数**random**可用来创建服从正态分布的$m \times n$随机矩阵。如果期望值v没有给定，

则令v=0。

```
function  A = Random(m,n,v)
% 返回一个方差为0、期望值为 v的m×n矩阵。
% 如果v没有给定，则使用v=0。
if nargin == 2, v = 0; end
A = randn(m,n) + v;
```

调用

```
A = Random(2,2,4); B = Random(2,2);
```

给出**A**元素的期望值为4，**B**元素的期望值为0。

■

因为细胞矩阵中的细胞可以是不同的数据类型，所以就可以将不同类型的输入、输出参数放入到细胞矩阵**varargin**和**varargout**中。示例如下：

■ **例**12.10

用一个函数来对任意多个向量计算每个向量的平均值、中位值和标准方差。可以使用下面的方法来计算：

```
function [varargout] = stat(varargin)

for i = 1:length(varargin)
  x = varargin{i};        % 取出输入参数
  y.medel = mean(x);      % 将结果保存到结构中
  y.median = median(x);
  y.std = std(x);
  varargout{i} = y;       % 将结果放入输出变量中
end
```

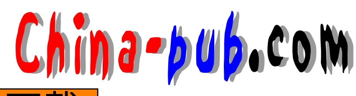
上面的程序中首先从细胞矩阵 **varargin**中挑选出一个输入的参量，然后计算出给这个参量的平均值、中位值和标准方差并将其结果保存到结构 **y**中。现在将这个结构放入到细胞矩阵**varargout**的一个细胞中。运行上面的程序能得到下面的结果：

```
a = [1 6 8 9];
b = [42 12 56 72 5 34];
[ares,bres] = stat(a,b)


ares =
     medel: 6
    median: 7
       std: 3.5590

bres =
     medel: 36.8333
    median: 38
       std: 25.5689
```

■



也许会将普通的输入、输出参数和 **varargin**和**varargout**混淆，但**varargin**和**varargout**在每一个参数列表的最后。下面是具有不同函数名的几个例子。

■ **例**12.11

函数test1接收参数*x*，然后接收任意多个附加的参数，并只返回一个输出变量：

```
function y=test1(x, varargin)
```

函数test2只接收一个输入参数。但是可以返回所需的输出参量和任意多个附加的输出参量：

```
function[y, varargout]=test2(x)
```

■

一个函数也可以有一个可选的返回参量。例如 bar(x,y)在向量**x**处画出向量**y**元素的图形，而[xx,yy]=bar(x,y)不画出图形，而只返回向量**xx**和**yy**。命令plot(xx,yy)也能画出与bar(x,y)相同的图形。有关bar更多的内容参见6.5节。

■ **例**12.12

下面的M文件**ngon.m**能计算出c^n=1的根作为缺省值，但也可定义一个复数 *z*作为可选的输入参量，那么就计算c^n=z的根。

这些根在复平面上定义了一个 *n*边形。如果不给出任何输出参量，就画出多边形。如果给出输出参数，就得到一个在复平面上定义多边形边角的复数向量。如果给出两个输出参数，就得到两个在平面上定义多边形的实数向量。

```
function [aa,bb] = ngon(n,z)

% 文件: ngon.m
% c^n=z的n个根是一个复平面上n边形的边角

% 对每种情况：0, 1, 2, 检查返回值的个数。

% 如果只有一个输入变量，则将z设为默认值1。

if nargin == 1
  z = 1;
end

%根为c=re+i*im, k=1:n

k = 1:n;
re = abs(z)*cos((angle(z)+(k-1)*2*pi)/n);
im = abs(z)*sin((angle(z)+(k-1)*2*pi)/n);
xx = [re re(1)]; yy = [im im(1)];

% 检查要求的输出参数的个数
% nargout == 0:
% 根据z的相角在灰色底面上画出n边形。

% nargout == 1:
```




```
%返回复数向量，以便plot(xx, yy)可在复平面上画出多边形的轮廓。
% nargout == 2:
%返回实数xx和yy向量，以便plot(xx, yy)画出多边形轮廓。

if nargout == 0
  patch(xx,yy,[abs(angle(z)/pi) abs(angle(z)/pi)...
                                       abs(angle(z)/pi)])
  axis('equal')
elseif nargout == 1
  aa = xx + yy*i;
else
  aa = xx;
  bb = yy;
end
```

在2.4节中介绍了命令angle，在14.2.11中介绍命令patch。为了说明ngon的功能，给出下面的命令：

```
subplot(2, 2, 1);  ngon(5);
```

ngon将画出左上角图形，这个方程的解是 $c^n=1$，$n=5$：

```
subplo(2, 2, 2);  cv=ngon(5, i);  plot(cv);  axis('equal')
```

右上角图形是在复平面上用方程式为 $c_n=i$、$n=5$的解画出的：

```
subplot(2,2,3); [rv1,rv2] = ngon(5,-1);
plot(rv1,rv2); axis('equal')
```

左下角图形是用方程 $c^n=-1$,$n=5$的解画出的多边形，和rv1+i*rv2相等：

```
subplot(2, 2, 4); ngon(5,-i);
```

右下角图形的多边形的角是方程 $c^n=-i$、$n=5$的解。

这些多边形如图12-4所示。subplot和plot的定义见第13章。

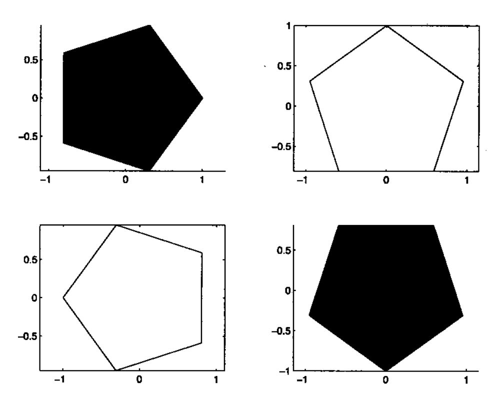
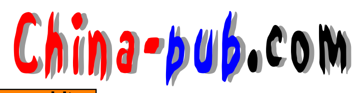


图12-4 用户定义的函数**ngon**生成的五边形

■

函数文件中的所有变量都是**局部变量**。因此，在一个函数文件中的变量与 MATLAB工作区中的同名变量是完全不同的变量，它们存在内存的不同位置。象所有的规则一样，这个规则有一个例外：全局变量可在MATLAB中使用。



一个全局变量在所有声明它为**global**的函数文件中都是可访问的。使用命令 who和 whos 可以知道哪些变量声明为全局变量。要清除全局变量，参见 2.3节。

通常变量的值在函数调用时会发生改变。如果要让变量的值不变，应该将它声明为 persistent。如何声明可参见例12.13。

■ **例**12.13

创建函数persdemo。

```
function persdemo(x)

if x
  persistent TIMESUSED;
  TIMESUSED
end;
if (exist('TIMESUSED','var') & ~isempty(TIMESUSED))
   TIMESUSED = TIMESUSED + 1;
 else
   TIMESUSED = 1;
end;

TIMESUSED

if TIMESUSED < 3
  disp('Keep on calling');
else
  disp('Type clear persdemo to clear persistent value');
end;
```

运行程序，可得：

```
persdemo(1)

TIMESUSED =
     []
TIMESUSED =
     1
Keep on calling
persdemo(1)

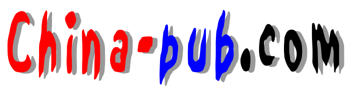
TIMESUSED =
     1
TIMESUSED =
     2
Keep on calling

persdemo(1)

TIMESUSED =
     2
TIMESUSED =
     3
Type clear persdemo to clear persistent value
```

以0作为参量调用可使**TIMEUSED**被解释成不能保持其值不变。

```
persdemo(0)
```




```
TIMESUSED =
     1
Keep on calling

persdemo(0)

TIMESUSED =
     1
Keep on calling
```

如果persdemo再次以**TIMEUSED**作为不变值来调用，就能得到：

```
persdemo(1)

TIMESUSED =
     3
TIMESUSED =
     4
Type clear persdemo to clear persistent value
```

要零化变量必须将其从内存中清除。

```
clear persdemo,persdemo(1)

TIMESUSED =
     []
TIMESUSED =
     1
Keep on calling
```

■

下面是控制M文件的命令。

命令集113　M文件的控制

`run filename`	运行命令文件**filename**，**filename**包括文件的全部路径和文件名。
`pause`	暂停M文件的运行，按下任意键后继续运行(见例13.19(c))。
`pause(n)`	暂停运行*n*秒后继续执行。这个暂停命令在显示大量图形时非常有用。
`pause off`	指示MATLAB跳过后面的暂停。
`pause on`	指示MATLAB遇到暂停时执行暂停命令。
`break`	终止`for`和`while`循环。如果在一个嵌套循环中使用该命令，只有内部循环被终止；参见12.2节。
`return`	结束M文件运行，MATLAB立即返回到函数被调用的地方。
`error(str)`	终止M文件的运行，并在屏幕上显示错误信息和字符串 str。
`errortrap state`	决定当有错误发生时是否停止运行。**state**的值可为`on`(此时捕获错误信息并继续执行)或`off`(发生错误时停止运行)。
`global`	声明变量为全局变量。全局变量能在函数文件中被访问，而不必包括在参数列表中。命令`global`后面是以空格分开的变量列表。

	声明为全局变量将保持其全局性直至工作区完全被清除掉或使用了clear global命令。
isglobal(name)	如果变量**name**是全局变量，则返回1；否则返回0。
keyboard	将键盘当成一个命令文件来调用。当给出一个内部的M文件，运行将被暂停，这样就可在MATLAB的命令窗口中给出命令。提示符*k*>>表示这种特殊状态。当执行一个M文件时，这是检查或改变参数变量的一个很好的方法，所有命令都可以在命令窗口中输入。当输入关键字return时，M文件将继续运行。如果在一个函数文件中调用keyboard，那么该函数的工作区和它的全局变量都可访问。命令keyboard在调试过程中很有用。
mfilename	返回正在运行的M文件名字符串，一个函数能用这个函数获得它自己的名字。
warning(message)	在字符串**message**中显示一条警告信息但不终止程序运行。
warning val	控制警告信息。val合法的值有： off　终止后面的警告信息。 on　将警告信息再次打开。 backtrace　显示造成警告的所在命令行。 debug　当发生警告时激活调试器。 once　每部分只显示一次与警告向下兼容的图形句柄。 always　显示所有的警告信息。
[vt,f]=warning	将当前警告状态vt和警告频率f作为字符串返回。

对于用户自定义的函数可以有子函数。这些子函数只能被其与M文件同名的主函数或在M文件中的其他函数所调用。在一个M文件中只能有一个主函数。在文件main.m中有一个函数结构及其子函数的示例：

```
function y=main(x)          % 主函数。
...                         % 程序语句。
z1=under1(x);               % 调用第1个子函数。
...                         % 程序语句。
y=under2(a);                % 调用第2个子函数。
...                         % 程序语句　。
function y=under1(x)        % 第1个子函数。
...                         % 程序语句。
function y=under2(x)        % 第2个子函数。
...                         % 程序语句。
[b1,b2]=under3(a1,a2);      % 调用第3个子函数。
...                         % 程序语句。
function y=under3(x1,x2)    % 第3个子函数。
...                         % 程序语句。
```

还有一类函数称为私有函数，这一类函数是放入一个叫private子目录中的M文件，私有函数只能被private直接上层目录中的函数调用。

当MATLAB在调用在M文件中的函数时，它首先查找子函数，再查找私有函数，最后在MATLAB的搜索路径中查找函数；见命令集 22。这就表明用户可以创建与 MATLAB函数同名的私有函数，并将其放入 **private**子目录中，这样程序就能对它们进行调用。同时，其他路径下的程序能调用和私有函数的同名的 M文件，但此时执行的是MATLAB的函数。

12.4 将函数作为参数传递给其他函数

在大部分高级语言中，如 Pascal或FORTRAN，能够创建一个通用函数 **F**，该函数是以另一个函数**f**作为参数的。在MATLAB中同样也能这样做，将函数的表达式或函数名包含在字符串**f**中。在函数**F**中，函数**f**能用`eval`或`feval`来求值。

命令`eval`(见5.1.4节)对包含MATLAB表达式的字符串求值，比如，这个字符串可包含数学表达式：

```
a = eval('sin(2*pi)') or
x = 2*pi; b = eval('sin(x)')
```

如果要用命令`val`，则字符串f中使用的变量必须要和**F**中的变量名字相同。

命令`feval`既可以对内建函数如 `sin`、也可以对保存在 M文件中的函数求值。调用`feval`的形式如下：

```
a = feval('sin',2*pi)
```

命令集114 求值函数

`feval(fcn,x1,···,xn)`	对字符串**fcn**给出的函数求值。参数*x*1, ··· ,**xn**是按出现的顺序传递给函数。`feval`命令通常在以其他函数为参数的函数内使用。
`[y1,y2,···]= feval(fcn,x1,···,xn)`	同上，但返回多个变量。

假设函数**fcn**带有元素操作运算符，这些运算符也就是 +， —， .*, ./ ,.\, .^。如果**x**是一个向量，则命令`feval(fcn,x)`也返回一个向量。如果在 **F**中使用`feval`，则将一个被赋予向量值的函数作为参数传给 **F**是没有问题的。如果在 **F**中使用了命令`eval`，并且`eval`直接作用于向量，那么，算术操作运算符必须使用在函数 **f**的参数字符串中。

■ **例**12.14

现在要编写MATLAB函数来得到$f(x)$的数值表，x在区间[a, b]内，步长为k。假定函数**f**是使用算术操作操作符定义的。

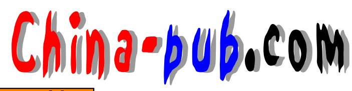
(a) 用`feval`：

输入参量：包含函数名的字符串 A，上下限a和b及步长k。

输出参量：有两列向量的矩阵 A，矩阵值为x值和$f(x)$。

下面的函数保存在文件 **Functabl.m**中：

```
function Y=Functab1(f, a, b, k)
% 在区间[a,b]中对一个标量函数求值, x(j)=a+j*k。
```




```
% 结果在表中，其中包含x值和f(x)。

x = a:k:b;
z = feval(f,x);
Y = [x;z]';
```

(b) 用eval。将命令字符串作为函数的输入参量就能得到相同的结果。使用 eval命令对这个命令字符串求值；参见5.1.4节。

输入参量：包含定义函数的命令字符串 A，上下限*a*和*b*及步长*k*。

输出参量：有两个列向量的矩阵 A，矩阵值为*x*值和*f*(*x*)。

下面的函数保存在文件**Functab2.m**中：

```
function Y=Functab2(f, a, b, k)
% 在区间[a, b]中对一个标量函数求值, x(j)=a+j*k。
% 结果在表中，其中包含x值 和f(x)。
x=a:k:b;
z=eval(f);              % f必须是x的函数。
Y=[x; z]';
```

假设想要得到函数*oneplusx*(x)=1+*x*的函数值表，*x*在区间[－1, 1]中。函数**oneplusx**保存在文件**oneplusx.m**中。下面的例子即为如何调用函数 **Functab1**和函数**Functab2**来创建这样的表。它们都给出相同的结果。

```
Tab = Functab1('oneplusx',-1,1,0.25)

Tab =
   -1.0000         0
   -0.7500    0.2500
   -0.5000    0.5000
   -0.2500    0.7500
         0    1.0000
    0.2500    1.2500
    0.5000    1.5000
    0.7500    1.7500
    1.0000    2.0000
```

另一种创建表格的方法为：

```
Tab = Functab2('oneplusx(x)',-1,1,0.25)

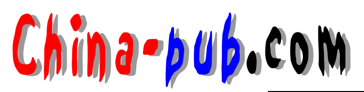
or

Tab = Functab2('1+x',-1,1,0.25)
```

■

12.5 结构

MATLAB能创建有多个域的结构 **structs**。在许多现代程序语言中都用到了结构，如C/C++中的结构和 Pascal中的记录，但它们之间也有一些差别。可以直接分配或使用 struct命令来创建一个结构。结构类型是可变的，所以结构的向量不必有相同的数据类型。但对于某些域要求其具有相同的数据类型；见例 12.15。对于整个结构来说，唯一的限制就是域只能包含标量或者固定维数的细胞矩阵。



为了从这些域中获取数据，应该在结构名和域名之间使用句点表达式，‘ . ’；见例12.15。下面对结构操作的函数都是可用的：

命令集115 有关结构的函数

`struct(f1,V1,f2, V2,...)`	返回带有域*f1* ,*f2*, …及其相应域值V1, V2, …的结构。参数V1, V2, …可以是相同大小的细胞矩阵或标量。
`fieldname(S)`	返回结构S中域名的列向量。
`getfield(S,f)`	返回结构S中域*f*的值。也可以写成`S.f`。
`isstruct(S)`	如果S是结构，则返回1；否则返回0。
`isfield(x)`	如果S是结构中的一个域，则返回1；否则返回0。
`setfield(S,f,v)`	设定结构S中域*f*的值为v，也可以写成`S.f=v`。
`rmfield(S,fvect)`	在向量**fvect**中返回无域的结构S。
`struct2cell(S)`	返回带有结构S中值的细胞矩阵。
`handle2struct`	将图形层次句柄转换成带有 **type**域的结构(对象类型，如**line**、**handle**、**properties**、**children**或**special**)。高级图像和句柄将在第14章中讨论。
`struct2handle`	将一个结构转换成图形层次句柄。其域与`handle2struct`中的相同。
`[out1,out2,...]= deal(in1,in2,...)`	将输入拷贝到输出，即`out1=in1`；`out2=in2`；可参见`helpdesk`中的示例。

■ **例**12.15

一个存储方程的结构，其名字和描述如下：

```
curve(1).name = 'Circle';
curve(1).function = '(x-a)^2 + y(-b)^2 = r^2';
curve(1).description =
'Circle with radius r centered in x = a, y=b';
```

使用`struct`在向量curve中创建其他元素：

```
curve(2) =
struct('name','line','function',2,'description','A two?')'
```

尽管在第1种情况下域*function*中的值是字符串，而在第2种情况下为整数，这种方法还是有效的。注意：**curve**是两个结构的向量。

■

12.6 对象

在MATLAB中，对象不同于结构，因为存在着将函数和对象联系起来的可能性，并且对象是根据类来创建的。在面向对象的上下文中，这些函数通常被叫做 **methods**(方法)。这样可以保护类中的成员变量，而只有类方法才能对它们进行访问。

命令集116 面向对象的函数

`class(object)`	返回对象**object**的类名。
`class(object,class, parent1,parent2,…)`	返回**object**作为**class**的变量。如果返回的对象要有继承属性，则应给定参数**parent1**, **parent2**,…。
`isa(object,class)`	如果**object**是**class**类型，则返回1；否则返回0。
`isobject(x)`	如果**x**是一个对象，则返回1；否则返回0。
`superiorto(class1, class2,…)`	当调用方法时，控制优先权的次序。如果要将一个类定义成`superiorto`，首先就用这种方法。
`inferiorto(class1, class2,…)`	当调用方法时，控制优先权的次序。如果要将一个类定义成`inferiorto`，最后用这种方法。
`methods class`	返回类**class**定义的方法名字。

为了创建一个类对象，就必须定义类的属性 **properties**，如创建一个类对象的模板。首先要创建一个目录。该目录必须与类同名但开头加上 @的记号(在VAX/VMS系统中使用$符号)，然后根据类模板需要一个能创建对象的构造函数 **constructor**。这就是根据类模板返回变量的函数(在一个**M**文件中)。

■ **例**12.16

要创建一个名为 **curve**的对象，首先要创建一个目录 **@curve**，然后在文件 **@ curve / curve.m**中创建构造程序。具体如下：

```
function l = curve(a)

% curve类的构造函数
% 1 = curve 创建并初始化一个curve对象
% 参数a可以是一个细胞数组，其中一个细胞是数学函数，另一个是说明或
另一个curve对象

% 数学公式必须和FPLOT要求的形式相同，参见FPLOT

% 如果没有传递参数，则返回包含x轴的一个对象

if nargin == 0   % 在此情况下为缺省的构造函数
        l.fcn = '0';
        l.descr = 'x axis';
        l = class(l, 'curve');

% 如果传递的参数是一个curve对象，则返回该对象的副本

elseif isa(a,'curve')
        l = a;
elseif (ischar(a{1}) & ischar(a{2}))
        l.fcn = a{1};
        l.descr = a{2};
        l = class(l, 'curve');
```




```
% 如果传递的参数是错误类型，则将给出错误信息

else
        disp('Curve class error #1: Invalid argument.')
end
```

■

例12.16中的line l=class(l,curve')给出了与类有关的变量。如果不考虑这个，则只返回一个不能对类方法访问的结构。利用方法才能使用对象的值，即，将与类连接并且分别在M文件中创建的函数放入类目录中。注意，对象是那些不同于其他面向对象语言参量中的一个，这里它的语法是object.method(参数))。

■ **例**12.17

根据这些描述，可以画出例12.16的曲线：

```
function p = plot(l,area)

% curve.plot 在area中画出函数curvel的图形
% area必须是一个1×4的矩阵，元素为XMIN XMAX YMIN YMAX
% 或一个1×2的向量，元素为XMIN XMAX

% 产生步长和一个带x值的向量

step = (limits(2)-limits(1))/40;
x = limits(1):step:limits(2);

% 画出函数图形

fplot(l.function, limits);
title(desciption)
```

运行类curve，如下：

```
parabola = curve({'x*x' 'A parabola'})

parabola=
curve=object: 1-by-1

plot(parabola,[-2 2])
```

结果如图12-5所示。

■

对象的值只对方法是有用的。这给程序员提供了一种检查输入的变量的途径，而结构无法提供这种途径。因此，需要一个对于类特定的方法，以便用户能够改变对象的域。

在MATLAB中，不同的运算符对于不同的类是不同的。为了重载运算符，就必须用运算符的名字创建一个方法，方法的代码指明运算符的功能。

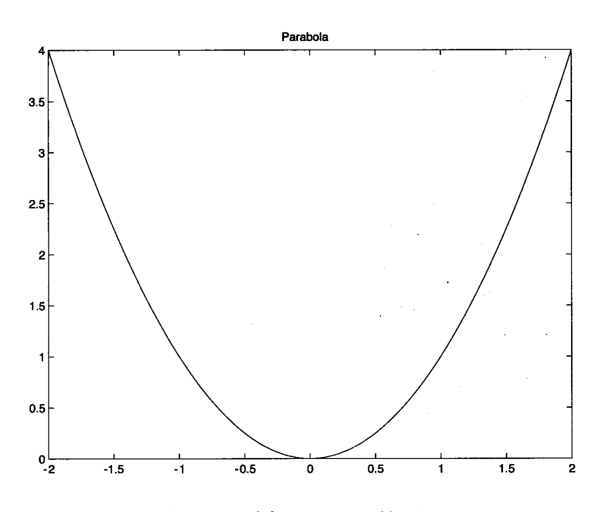


图12-5　对象parabola的图形

■ **例**12.18

为了能使两个**curve**类的对象相加，可以在目录@**curve**下创建一个M文件来重载加法运算符。注意，该方法对于对象的数据是完全可访问的。

```
function ltot = plus(l1,l2)

% 将曲线L1和L2相加

function = strcat(l1.function,' + ',l2.function);
description = strcat(l1.description,' plus ',l2.description);
ltot = curve({function description});
```

这样就能输入`l1+l2`或`plus(l1,l2)`，将两条曲线l1和l2相加。

■

输入`help operatorname`就能知道运算符到底是如何进行工作的，以及哪些对象有运算符重载。运算符定义在命令集117中。

命令集117　对象的运算符重载

`plus(a,b)`	表示**a**+**b**的函数。
`minus(a,b)`	表示**a**－**b**的函数。
`uplus(a)`	表示+**a**的函数。
`uminus(a)`	表示－**a**的函数。
`times(a,b)`	表示**a**.****b**的函数。
`mtimes(a,b)`	表示**a*****b**的函数。
`rdivide(a,b)`	表示**a**./**b**的函数。
`ldivide(a,b)`	表示**a**.**b**的函数。
`mrdivide(a,b)`	表示**a**/b的函数。
`mldivide(a,b)`	表示**a****b**的函数。
`power(a,b)`	表示**a**.^{b}的函数。

`mpower(a,b)`	表示 a^b 的函数。
`lt(a,b)`	表示**a**<**b**的函数。
`gt(a,b)`	表示**a**>**b**的函数。
`le(a,b)`	表示**a**<=**b**的函数。
`ge(a,b)`	表示**a**>=**b**的函数。
`ne(a,b)`	表示**a**=**b**的函数。
`eq(a,b)`	表示**a**= =**b**的函数。
`and(a,b)`	表示**a**&**b**的函数。
`or(a,b)`	表示**a**|**b**的函数。
`not(a)`	表示**a** 的函数。
`colon(a,b)`	表示**a**:**b**的函数。
`colon(a,s,b)`	表示**a**:s:**b**的函数。
`transpose(a)`	表示**a**´的函数。
`ctranspose(a)`	表示**a**. 的函数。
`display(a)`	表示**a**的函数。
`horzcat(a,b,···)`	表示[**a b**···]的函数。
`vertcat(a,b,···)`	表示[**a**;´**b**;···]的函数。
`subsref(a,i)`	表示 $\mathbf{a}(\mathbf{i}_1, \mathbf{i}_2,\cdots,\mathbf{i}_n)$ 的函数，当重新定义该函数时，命令 `substruct`非常有用。
`subsasgn(a,i,b)`	表示 $\mathbf{a}(\mathbf{i}_1,\mathbf{i}_2,\cdots,\mathbf{i}_n)=\mathbf{b}$ 的函数。
`subsindex(a,b)`	表示**b**(**a**)的函数。

当函数有一个重载运算符时，而用户又希望能在 `builtin`命令中使用普通运算符。

命令集118　跳过重载运算符

`builtin(fcn,x1,x2,···)`	用参数*x*1,*x*2, ···对内建函数**fcn**求值，而不是使用重载运算符。

在MATLAB中的类能从其他类中继承属性，这方面的使用已超出本书范围，推荐从 C++的书中可以找到这方面的有关信息。要创建能继承属性的对象，可以在 `class`命令中对其父对象进行设置。

12.7　调试和计时

MATLAB中有些命令对调试M文件很有用。可以是在调试过程中寻找错误，设置和清除断点，逐行运行M文件，或在不同的工作区检查变量。所有的调试命令都是以字母 `db`开头，已用过的命令`dbtype`(见2.9节)，就能生成带行数的程序列表。

所有设置、清除和列出断点的命令在下面的命令集 119中给出。

命令集119 断点

dbstop in fname	在M文件**fname**的第一可执行程序上设置断点。
dbstop at r in fname	在M文件**fname**的第*r*行程序上设置断点。如果第*r*行程序是不可执行的，则程序会在运行行可执行程序后停止。
dbstop if v	当遇到条件*v*时，停止运行程序。当发生错误时，条件*v*可以是error，当发生NaN或inf时，也可以是naninf/infnan。
dstop if warning	如果有警告，则停止运行程序。
dbclear at r in fname	清除文件**fname**的第*r*行处断点。
dbclear all in fname	清除文件**fname**中的所有断点。
dbclear all	清除所有M文件中的所有断点。
dbclear in fname	清除文件**fname**第一可执行程序上的所有断点。
dbclear if v	清除第*v*行由dbstop if v设置的断点。
dbstatus fname	在文件**fname**中列出所有的断点。
mdbstatus	显示存放在dbstatus中用分号隔开的行数信息。

这些命令与下面列出的命令用来跟踪和控制 M文件运行是非常有用的。在调试中使用try/catch结构也是非常有用的；参见12.1节。例如，如果使用try/catch来解例5.8(a)的问题，可以得到比用eval来解更多的通解。

命令集120 运行控制命令

dbstep	运行M文件的下一行程序。
dbstep n	执行下*n*行程序，然后停止。
dbstep in	在下一个调用函数的第一可执行程序处停止运行。
dbcont	执行所有行程序直至遇到下一个断点或到达文件尾。
dbmex	调试MEX文件的命令，见 15.2.1节和15.3.1节。在Windows中或Macintosh系统中，这个命令是无效的。
dbquit	退出调试模式。

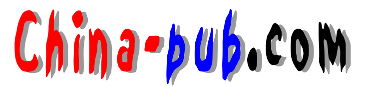
进行程序调试，要调用带有一个断点的函数。当 MATLAB进入调试模式时，以 K作为该状态的提示符：*K*>>。最重要的区别在于现在能访问函数的局部变量，但不能访问 MATLAB工作区中的变量。以函数**Factab.m**来举例说明，该函数能产生1!, … ,n!的阶乘表。

■ **例12.19**

首先列出函数**Factab.m**的行数：

```
dbtype Factab

1    function Tab = Factab(n)
2    %
3    %生成一个1!, …,  n!的阶乘表
4
```




```
5     numbers = 1:n; facts = [];
6
7     for i = numbers
8       facts = [facts factorial(i)]
9     end
10
11    Tab = [numbers' facts'];
```

这个函数调用了函数factorial，程序如下：

```
dbtype factorial

1     function p = factorial(nn)
2     %
3     %计算nn的阶乘
4
5     if ( nn == 0 )
6       p = 1;
7     else
8       p = nn*factorial(nn-1);
9     end
```

开始调试程序，在函数第一行可执行程序处设置一个断点，然后调用该函数。注意提示符中字母K。

```
dbstop in Factab                % 在Factab中设置断点
Table = Factab(5);              % 调用Factab并调试
5    numbers = 1:n;
K>> dbstep                      % 执行一行程序
6    facts = [];
K>> numbers                     % 返回程序行数

numbers =

     1     2     3     4     5

K>> numbers = [numbers 6]       % 用数字6扩充向量

numbers =

     1     2     3     4     5     6

K>> dbstop 12                   % 在第12行设置断点
K>> dbcont                      % 继续执行到下一个断点
12   Tab = [numbers' facts'];
K>> dbquit                      % 退出调试

dbstatus Factab                 % 列出所有用过的断点
Breakpoints for Factab are on lines 5, 12.
```

■

函数调用另一函数可认为是嵌套的函数调用。MATLAB使用栈来跟踪工作区和函数中的变量，下面命令集中的命令就可用来在嵌套函数的工作区中进行切换。

命令集121 **切换工作区**

dbstep in	如果下一可执行程序行是函数调用，则跟踪函数。
dbup	切换到调用函数的工作区以检查变量。
dbdown	切换回被调函数的工作区。
dbstack	显示嵌套函数调用的栈。

■ **例**12.20

再次使用函数**Factab**；见例12.19。都以在**Factab**和**factorial**中设置断点开始：

```
dbstop Factab                   % 在Factab中设置断点
dbstop factorial                % Factroial中设置断点
Factab(3);                      % 调用Factab
5    numbers = 1:n;
K>> dbcont                      % 跟踪到下一个断点
5    if ( nn == 0 )
K>> dbstack                     % 跟踪进入factorial函数
In /home/aw/BOOK/factorial.m at line 5
In /home/aw/BOOK/Factab.m at line 8
K>> who                         % 显示当前值

Your variables are:

nn         p

K>> dbup                        % 切换到调用工作区
In workspace belonging to /home/aw/BOOK/Factab.m.
K>> who                         % 显示当前变量

Your variables are:

Tab        facts     i         n         numbers

K>> dbdown                      % 返回到factorial工作区
In workspace belonging to /home/aw/BOOK/factorial.m.
K>> dbquit                      % 退出调试
```

要调试命令文件，必须使用命令keyboard。MATLAB的特定调试命令只能用在调试函数文件中。
■

为了编写有效的程序，需要一种工具来计算哪一部分程序所需时间最多。 profile命令就是这样的工具：

命令集122 **M文件的计时**

profile choice	M文件的计时命令。参数**choice**可以为下面情况之一：	
	filename	给M文件**filename**计时；在同一时间里只能对一个文件计时。

	on,off	打开或关闭指定M文件的计时器。
	reset	清除程序评述器的计时数据。
	report	显示计时报告。
	report n	显示所花时间最多的*n*行程序。
	report frac	显示至少使用部分时间的程序行。时间的*frac*，*frac*的值必须在区间[0，1]中。
	done	结束定时。
info=profile	返回结构**info**，用于计时器数据的图形显示。它包括下面字段：	
	info.file	存放文件名，包含被计时的 M文件的完全路径。
	info.function	包含函数名。
	info.interval	包含计时区间。
	info.count	包含带有计时数据的向量。
	info.state	包括计时工具的状态: on 或off。
profsumm choice	创建M文件的评述摘要。**choice**的有效值为：	
	n	显示程序中使用时间最多的*n*行程序。
	frac	显示程序中至少使用时间 *frac*的程序行，*frac*的值必须在0和1之间。
	str	如果程序行中有字符串**str**，则报告。
	profsumm	命令也可以使用保存在结构**info**中的信息。

■ **例**12.21

同时对两个不同的程序进行计时，而这两个程序都完成相同的任务，通过计时来判断哪个程序将效率更高。第一个程序在**particle.m**中，如下：

```
% 随机路径。一个质点从原点开始，每一步在8个方向任意地走半个单位

%disp('Give the number of steps')   % 步数
%n = input('>>>');
n = 500;

x = cumsum(rand(n,1)-0.5);           % 任取x值
y = cumsum(rand(n,1)-0.5);           % 任取y值

clf;                                 % 清除图形窗口
plot(x,y);                           % 画出路径
hold on;                             % 保持当前图形
plot(x(1),y(1),'o',x(n),y(n),'o');   % 标出start/finsh
axs   = axis;                        % 取min和max
scale = axs(2) - axs(1);             % 计算刻度值

text(x(1)+scale/30,y(1),'Start');    % 在Start和Finish
```

```
text(x(n)+scale/30,y(n),'Finish'); % 右边写出文本

hold off;                          % 删除图形

xlabel('x'); ylabel('y');
title('Random walk')
```

另一个程序在文件particleBad.m中：

```
% 随机路径。一个质点从原点开始，每一步在8个方向任意地走半个单位

%disp('Give the number of steps')  % 步数
%n = input('>>>');
n = 500;
%x = cumsum(rand(n,1)-0.5);        % 任取x值
%y = cumsum(rand(n,1)-0.5);        % 任取y值

x(1) = rand(1,1)-0.5;
y(1) = rand(1,1)-0.5;

for i = 2:n
  x(i) = rand(1,1)-0.5 + x(i-1);
  y(i) = rand(1,1)-0.5 + y(i-1);
end

clf;                               % 清除图形窗口
plot(x,y);                         % 画出路径
hold on;                           % 保持当前图形
plot(x(1),y(1),'o',x(n),y(n),'o'); % 标出start/finish

axs   = axis;                      % 取min和max
scale = axs(2) - axs(1);           % 计算刻度值

text(x(1)+scale/30,y(1),'Start');  % 在Start和Finish
text(x(n)+scale/30,y(n),'Finish'); % 右边写出文本

hold off;                          % 删除图形

xlabel('x'); ylabel('y');
title('Random walk')
```

给出下面的命令：

```
profile particle; particle; profile report; profile off;
profile particleBad; particleBad; profile report; profile off;
```

结果为：

```
Total time in "particle.m": 0.22 seconds
100% of the total time was spent on lines:
        [13 25 12 26 17 15]

              11:
0.03s, 14%    12: clf;
```




```
                    % 清除图形窗口
0.11s, 50%    13: plot(x,y);
                    % 画出路径
              14: hold on;
                    % 保持当前图形
0.01s,  5%    15: plot(x(1),y(1),'o',x(n),y(n),'o');
                    % 标出start/finish
              16:
0.02s,  9%    17: axs   = axis;
                    % 取min和max
              18: scale = axs(2) - axs(1);
                    % 计算刻度值
              24:
0.03s, 14%    25: xlabel('x'); ylabel('y');
0.02s,  9%    26: title('Random walk')

Total time in "particleBad.m": 0.57 seconds

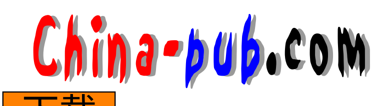
98% of the total time was spent on lines:
        [16 15 21 20 33 34 31 25 23 22]

              14: for i = 2:n
0.15s, 26%    15:   x(i) = rand(1,1)-0.5 + x(i-1);
0.18s, 32%    16:   y(i) = rand(1,1)-0.5 + y(i-1);
              17: end

              19:
0.04s,  7%    20: clf;
                    % 清除图形窗口
0.10s, 18%    21: plot(x,y);
                    % 画出路径
0.01s,  2%    22: hold on;
                    % 保持当前图形
0.01s,  2%    23: plot(x(1),y(1),'o',x(n),y(n),'o');
                    % 标出start/finish
              24:
0.01s,  2%    25: axs   = axis;
                    % 取min和max
              26: scale = axs(2) - axs(1);
                    % 计算刻度值
              30:
0.01s,  2%    31: hold off;
                    % 删除图形
              32:
0.03s,  5%    33: xlabel('x'); ylabel('y');
0.02s,  4%    34: title('Random walk')
```

这样就能知道程序所使用的时间，并且知道哪一行程序所花时间最多。如果要立即画出最新的profile运行结果，可以输入命令：

```
t = profile
```




```
t =
        file: '/home/matlab/VER5/kapitel12/particleBad.m'
    interval: 0.0100
       count: [33x1  double]
       state: 'off'

pareto(t.count)
```

在6.5节和13.1节中介绍了命令pareto。图12-6中的x轴上的棒形图为行程序运行所需的时间。y轴的左边是百分之一秒为单位的运行时间，右边是其所占全部执行时间的百分比。图上的线条表示总运行时间。

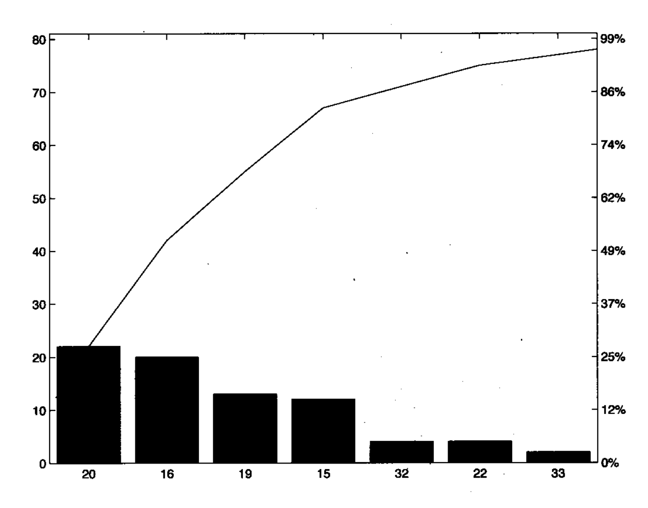


图12-6 particleBad.m计时的图形表示

■

第13章　图形和声音

MATLAB拥有大量简单、灵活、易用的二维和三维图形命令，并且用户可以在 MATLAB程序中加入声音效果。许多图形命令都在 MATLAB所带的演示程序中给出。还未用过 demo命令的用户不妨马上试一试。

图形命令分两章来介绍。本章只介绍基本的高级命令，在第 14章中将详细介绍高级图形，并着重介绍具体的低级控制。

13.1　二维图形

将数对排序的一种方法是使用 plot命令。该命令可以带有不同数目的参数。最简单的形式就是将数据传递给 plot，但是线条的类型和颜色可以通过使用字符串来指定，这里用 str表示。表13-1列出了在这个字符串中允许使用的线条类型和颜色。线条的缺省类型是实线型。注意：下面的命令列表中A表示一个$m \times n$的矩阵。

命令集123　绘图命令

`plot(x,y)`	对向量**x**绘制向量**y**。以x为横坐标，y为纵坐标，按照坐标(x_j，y_j)的有序排列绘制曲线。
`plot(y)`	以j为横坐标，y_j为纵坐标，绘制(j, y_j)的有序集合的图形。
`plot(z)`	以横轴为实轴，纵轴为虚轴，绘制(real(z_k),imag(z_k))的有序集合的图形。这样，复数z_k就在复平面上。
`plot(A)`	绘制矩阵**A**的列对它下标的图形。对于$m \times n$的矩阵**A**，有n个含有m个元素的数对，或是n条有m个点曲线，且这n条曲线均采用颜色监视器上不同的颜色绘制而成。
`plot(x,A)`	绘制矩阵**A**对向量**x**的图形。对$m \times n$的矩阵**A**和长度为m的向量**x**，绘制矩阵**A**的列对向量**x**的图形。如果**x**的长度为n，则绘制矩阵**A**的行对向量**x**的图形。向量**x**可以是行向量也可以是列向量。
`plot(A,x)`	对矩阵**A**绘制向量**x**的图形。对于一个$m \times n$的矩阵**A**和一个长度为m的向量**x**，对矩阵**A**的列绘制向量x的图形。如果**x**的长度为n，则对矩阵**A**的行绘制向量**x**的图形。向量**x**可以是行向量也可以是列向量。
`plot(A,B)`	对矩阵**A**的行绘制矩阵**B**的列的图形。如果**A**和**B**都是$m \times n$的矩阵，将绘制n条由m个有序对连成的曲线。
`plot(···,str)`	使用字符串**str**指定的颜色和线型进行绘图。表 13-1列出了**str**可以取的值。
`plot(x1,y1,str1,`	用字符串**str1**指定的颜色和线型对**y1**绘制**x**1的图形，用字符串

x2,y2,str2,...)	**str2**指定的颜色和线型对 **y2**绘制**x2**的图形…。每组参数值可以采用上述除复数值以外的任何一种形式。 **str1, str2**…可以省略，此时，MATLAB自动为每条曲线选择颜色和线型。
[1,f,p,errorm]=colstyle(str)	返回**str**中不同部分的值。其中**l**代表线型，**f**代表颜色，**p**代表点的类型, **errorm**用来保存系统错误信息。

表13-1　点类型、线类型与颜色

点 类 型		线 类 型	
.	点	-	实线
*	星号	--	虚线
square	正方形	—	点划线
diamond	菱形	:	点线
pentagram	五角星形	none	无线
hexagram	六角星形		
		颜色	
none	无点	g	绿色
o	o	m	品红色
+	+	b	蓝色
x	×	c	灰色
<	顶点指向左边的三角	w	白色
>	顶点指向右边的三角	r	红色
^	正三角	k	黑色
v	倒三角	y	黄色

通过将字符串str作为一个参数传递给plot，可以指定图形的颜色和线型。表 13-1列出了允许的值和它们代表的意义。这些参数可以组合起来使用，例如，‘ y+ ’表示一个黄色的加号，而‘ b—— ’表示一个蓝色的虚线。如果将要画的是几组数据，但是没有指定线型，系统将会自动按照表13-1赋予它们从黄到黑各种不同的颜色线型。

符号的大小、线条的粗细等也同样可以更改；可参见例 13.1(g)或14.2节。

■ **例13.1**

(a) 用下列数据来绘制图形：

```
x = [—4  -2  0  1  3  5]
y  =  [16  4  0  1  9  25]
```

命令plot(x,y)产生的结果如图13-1所示。

(b) 在MATLAB中，能很容易地画出点：

```
x = —pi:0.05:pi；
plot(x,sin(x)*cos(x),' o')；
```

产生的结果如图13-2所示。

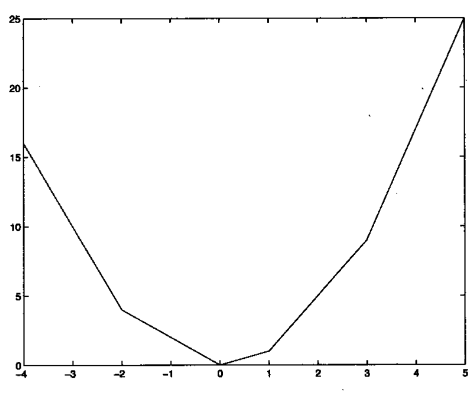


图13-1 向量**y**对向量**x**绘图

图13-2 用小圆圈绘制函数sin*x*×cos*x*

(c) 在同一幅图中可以同时绘制多个函数：

```
x = 0:0.1:2;
A = [sin(pi*x);0.5+0.5*x];
plot(x, A);
```

产生的结果如图13-3所示。

(d)可以通过交换参数位置来交换坐标轴。对图 13-3和图13-4进行比较：

```
x = 0:0.1:2;
A = [sin(pi*x);0.5+0.5*x];
plot(A, x);
```

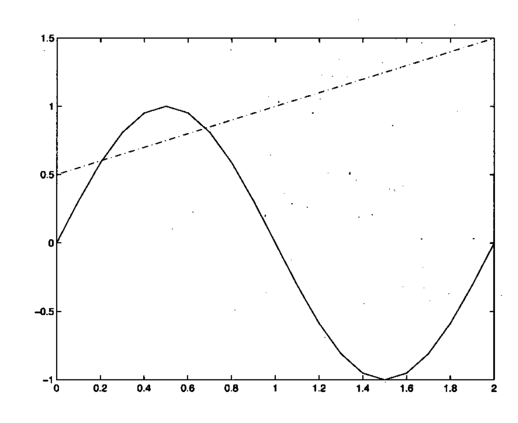




图13-3 矩阵**A**对向量**x**绘图

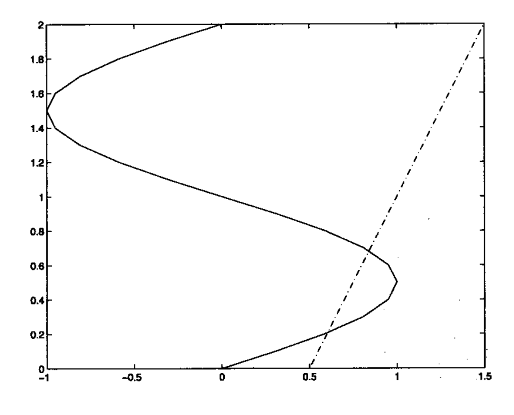
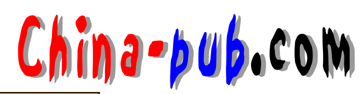


图13-4 向量**x**对矩阵**A**绘图

(e) plot命令对复数矩阵同样适用。

```
clear i;                        % 保证i是复数
r = linspace(0,2);              % 创建向量r
theta = linspace(0,10*pi);      % 创建角向量
[x, y] = pol2cart(theta,r);     % 将弧度坐标
z = x+i*y;                      % 转化成复数向量
plot(z)                         % 对z绘图
```

结果如图13-5所示。注意：还可以用命令 polar、quiver、feather、compass和rose来对复数绘图；参见13.2节。



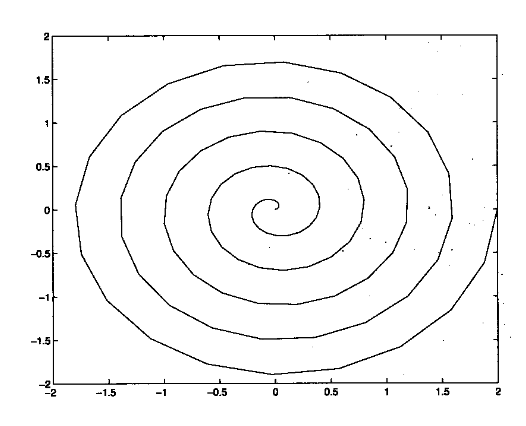


图13-5　复向量z代表一个螺旋线。

(f)下列命令形成文件expotest.m：

```
% 程序执行前应先设定下列
% 参数：n, a, b
% 点数：n.
% 区间：[a, b]

x  = [];
e1 = []; e2 =[]; e3 = []; e4 = []; %  清除  e1-e4

for i = 1:n
  xx    = a + (b-a)*(i-1)/(n-1);
  x(i)  = xx;
  e1(i) = exp(-(xx^2));
  e2(i) = xx^2*exp(-(xx^2));
  e3(i) = xx*exp(-(xx^2));
  e4(i) = exp(-xx);
end
```

尽管下列代码将会产生同样的结果，但它的效率更高，易读且不易产生错误。

```
x = linspace(a,b,n);

e1 = exp(-x.^2);
e2 = (x.^2).*exp(-x.^2);
e3 = x.*exp(-x.^2);
e4 = exp(-x);
```

下列语句：

```
n = 50;
a = 0;
b = 3;
expotest
plot(x,e1,x,e2,x,e3,x,e4);
```

将产生图13-6(左)所示的图形。而

```
plot(x,e1,'+',x,e2,'*',x,e3,'o',x,e4,'x');
```

将产生图13-6(右)所示的图形。

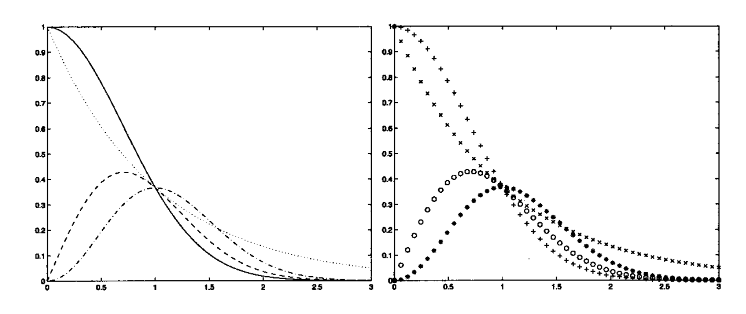


图13-6　用标准符号(左)和用户指定的符号(右)对指数函数绘图

(g) 假设已定义了与(f)中相同的变量。现在要来改变线条的粗细：

```
hold on;
plot(x,e1,'LineWidth',1);
plot(x,e2,'LineWidth',2);
plot(x,e3,'LineWidth',3);
plot(x,e4,'LineWidth',4);
hold off;
```

命令hold on用来保持当前图形，使得可以在同一幅图中绘制多个图形，而hold off用来关闭图形的；可参见命令集130。其中，曲线e1线条最细，e4线条最粗，如图13-7所示。

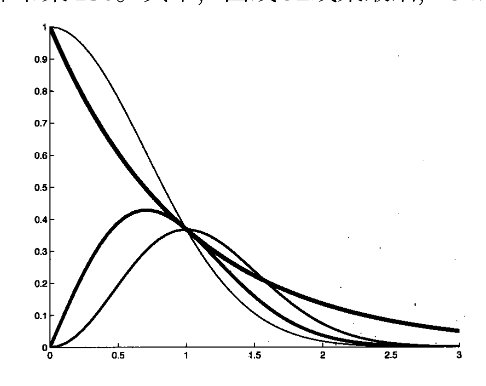


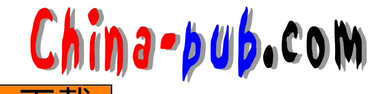
图13-7　用不同的粗细线条绘制的指数函数图形

■

在MATLAB中可以使用errorbar命令绘制数据的误差条形图。该命令的用法与plot命令完全类似，只是要同时赋予每个点一个误差限。

命令集124　误差条形图

errorbar(x,y,e,*str*)	绘制向量y对x的误差条形图。误差条对称地分布在 y_i的上方和下方，长度为 e_i。字符串**str**决定其颜色和线型，参见表13-1。参考命令集123中的命令plot。
errorbar(x,y,l,u,*str*)	绘制向量y对x的误差条形图，误差条分布在y_i上方的长度为u_i,下方的长度为l_i。字符串**str**选项决定其颜色和风格。



■ **例**13.2

假定误差限为15%，下面的程序将产生一系列数字，并生成该列数据的误差条形图。

```
x = linspace(0,10,50);          % 创建一系列值
y = exp(sin(x));                % 创建数据

delta = 0.15*y;                 % 计算15%的误差限
errorbar(x,y,delta);            % 绘出误差条形图
```

运行后可给出图13-8所示的图形。

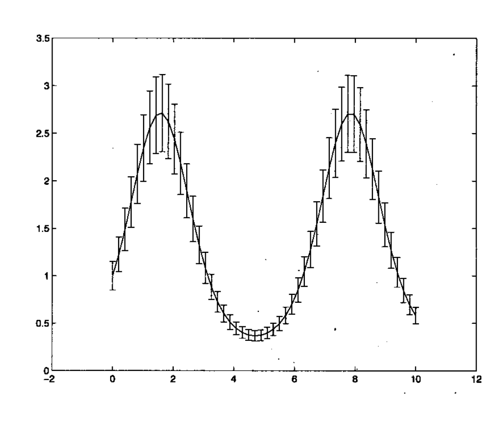


图13-8 函数$e^{\sin x}$的误差条形图

■

使用comet命令可以绘制**彗星图形**。彗星图形是一个动态的绘图过程。comet3命令可用来绘制三维彗星图形；参见13.5节。

命令集125 彗星图形

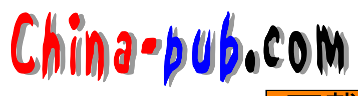
comet(x,y)	绘制向量y对向量x的彗星轨线。如果只给出一个向量，则用该向量对其下标值绘图
comet(x,y,l)	绘制慧长为*l*∗length(*y*)的彗星轨线，*l*的缺省值为0.1
comet	绘出一个慧星图形的例子

输入doc comet可得更多的信息。

MATLAB中有些函数可以用来改变图形的外观。

命令集126 其他绘图命令

area(x,y)	和plot命令一样，但是将所得的曲线下方即曲线与横轴之间的区域填充颜色。
area(x,A)	矩阵**A**的第一行对向量**x**绘图，然后依次是下一行与前面所有行值的和对向量**x**绘图。每个区域有各自的颜色。
area(y)	等价于x=1:size(y,1)
area(···,'Property',	为area创建的带下划线的补片对象设定属性；参见第14章。



Value,⋯)	
barh(x,A,*format*)	对$m\times n$矩阵绘制含有m组、每组n个柱形条的直方图。字符串**format**可以是颜色类型或字符串‘stacked’。‘stacked’表示将n个柱形条的值叠加在另一条上。
barh(A)	和barh命令一样，但是令x=1:m。
ezplot(f,*xmin*,*xmax*)	绘制函数**f**在区间[*xmin*, *xmax*]上的图形。如果省略*xmin*和*xmax*参数，区间将大概取在－2π～2π之间。由于ezplot命令使用算法来判断该函数变化显著的区间，因此区间的选取是不固定的。
pareto(y,x)	按降序绘制**y**中各分量的柱形图。可以给定向量**x**并且应该包含**x**轴的下标。如不给定，则将使用向量**y**中各元素的下标，同时，pareto命令还能对由各元素累积和形成的向量绘制曲线。
pie(x,*explode*)	绘制向量**x**的饼图。如果sum(x)<=1，则将给出一个不完全的饼图。向量**explode**与向量x的维数相同，并且**explode**中不为零的元素所对应的相应部分将从饼图中独立出来。
scatter(x,y,*size*,*color*)	以具有相同维数的向量**x**、**y**所确定的点为圆心，*size*(以点为单位)为半径绘制圆。圆的颜色由**color**确定，可以是向量、矩阵或颜色字符串。参见helpdesk可得更多信息。
plotmatrix(X,Y)	绘制X的列对Y的列的分散矩阵图形。
plotmatrix(X)	和plotmatrix(X,X)一样，但是在对角线上画出柱状图。
[H, AX, BigAx ,p]= plotmatrix(⋯)	返回整个图形的句柄**H**矩阵。矩阵**AX**包含单个子坐标系的句柄，**BigAx**包含的是大坐标系的句柄。柱状图的句柄保存在**P**中。留下**BigAx**作为当前句柄(如被axes使用)
plotyy(x1,y1,x2,y2,*fun1*, *fun2*)	**y1**按左侧y轴的刻度对**x1**绘图，**y2**按右侧y轴的刻度对**x2**绘图。若缺省参数**fun1**和**fun2**，则结果与使用plot命令相同。参数**fun1**和参数**fun2**可以是类似‘semilogx’,‘loglog’等的字符串。不同的函数绘图可参见第13.2节。

■ **例**13.3

用如下的方法可以在同一副图中绘制不同尺寸的图形。

```
% plotyy 演示
% 定义数据
x = 0:0.25:4;
y = exp(x);

clf reset;          % Plotyy对坐标轴上定义的数据很敏感
```

```
plotyy(x,y,x,y,'plot','semilogy')
hold on;
title('Plotyy')
ylabel('Linear scale')
```

产生结果如图13-9所示，不幸的是plotyy与其他命令一样会产生一些问题。例如：legend只适用于一个坐标轴。通过 ftp://ftp.mathworks.com/pub/tech-support/library/graphics/plotyy.m 可以获得plotyy.m文件。该文件给出了为所有坐标轴定义标识符的可能情况。

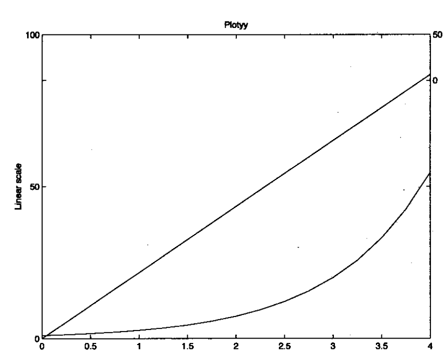


图13-9　借助plotyy 命令在同一幅图中用plot 和semilogy 绘制数据图形

■

■ **例**13.4

使用area命令，MATLAB可以绘制点的累积图形。

```
% 命令的演示
x = 0:10;
A = [ sin(x);x;(x/3).^2 ]';

clf;
areahandle = area(x,A)
hold on
title('Area plot')
legend(areahandle,'sin(x)','sin(x)+x','sin(x)+x+(x/3)^2',2)
```

结果如图13-10所示。

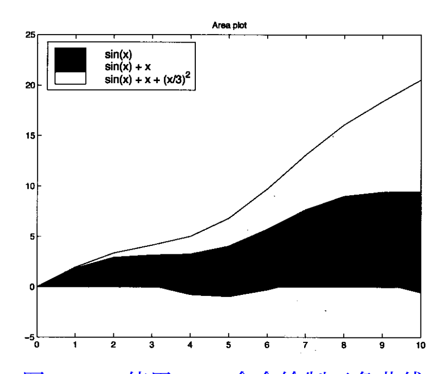
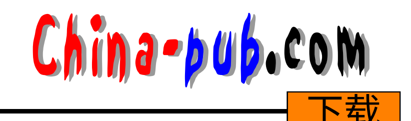


图13-10　使用area 命令绘制三条曲线

■

命令fplot可以绘制出标准的MATLAB和用户自定义的函数图形。区间范围和函数名字



可以作为参数给出。

命令集127 函数图形

fplot(fcn,lim,str)	绘制由字符串fcn指定的函数图形。这可以是标准函数，也可以是用户在M文件fcn.m中自定义的函数，但不允许是内联函数。向量lim=[x_{min} x_{max}]给出绘图区间范围。该向量也可以包含四个元素，后两个参数用来表示y轴的区间，即lim=[x_{min} x_{max} y_{min} y_{max}]。如果字符串str传递给fplot，则可以根据表13-1来改变图形的线型和颜色。
fplot(fcn,lim,str, tol)	同上所述进行绘图，但是带有一个小于 tol的相对误差

注意，使用fplot绘制所谓的内联函数是不可能的。

■ **例**13.5

用下面的语句来绘制sinx^2图形

```
fplot('sin(x.^2)',[0,10]);
```

将得到如图13-11所示的图形。

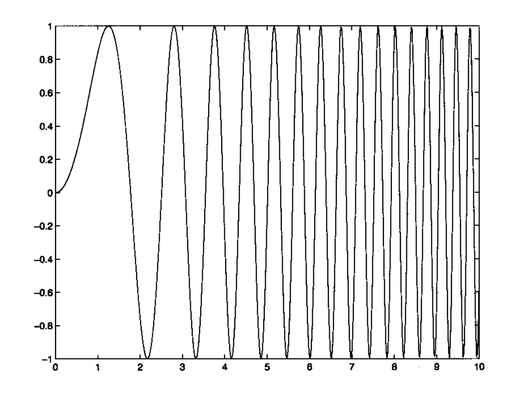


图13-11 使用fplot 命令绘制函数sinx^2的图形

■

13.2 在其他坐标系和复平面上绘图

plot命令使用的是笛卡儿坐标系，其实，使用其他的坐标系也是可以的。字符串参数 str可以传递给下列所有的命令，以确定绘图的颜色和线型 (参见表13-1)。

命令集128 在其他坐标系中绘图

polar(theta,r)	在极坐标中绘图。向量theta的元素代表弧度参数，向量r代表从极点开始的长度。
semilogx(x,y)	在半对数坐标系中绘图， x轴用以10为底的对数刻度标定。这类

	似于`plot(log10(x),y)`,但是对于$\log_{10}(0)$不能给出警告信息。
`semilogy(x,y)`	在半对数坐标系中绘图，y轴用以10为底的对数刻度标定。这类似于`plot(x,log10(y))`,但是对于$\log_{10}(0)$不能给出警告信息。
`loglog(x,y)`	在对数坐标系中绘图。两个坐标轴均用以 10为底的对数刻度标定。这类似于`plot(log10(x),log10(y))`,但是对于$\log_{10}(0)$不能给出警告信息。

参见第2.4节中关于更改坐标系的命令。

■ **例**13.6

(a) 在半对数刻度坐标系中绘图与在通常的笛卡儿坐标系中用 `plot`命令绘图一样容易。

```
x=linspace(0,7);                % 创建x值
y=exp(x);                       % 创建y值
subplot(2,1,1);plot(x,y);       % 绘制通常图形
subplot(2,1,2);semilogy(x,y);   % 绘制半对数刻度曲线
```

通过使用`subplot`命令可以在一个图形窗口中绘制多个小图形；见第 13.3节。执行上述命令，可以得到图13-12所示的图形。

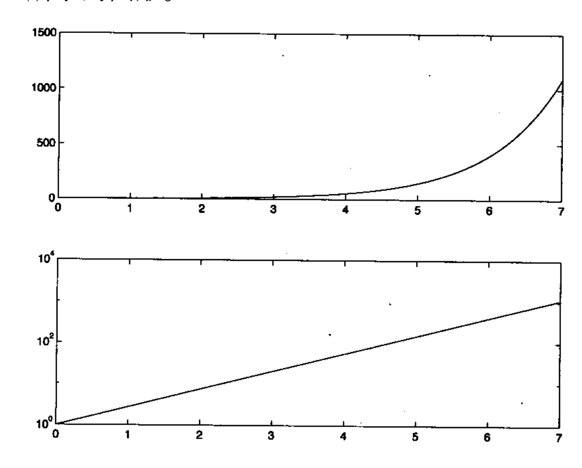


图13-12 在普通坐标系(上图)和y轴对数刻度坐标系(下图)中绘制指数函数

(b) 为了在极坐标系中绘制曲线，可以使用 `polar`命令。函数

$$r = e^{\cos t} - 2\cos 4t + \left(\sin\frac{t}{12}\right)^5$$

描绘的是一条复平面上的曲线。这里介绍绘制这条曲线的两种方法。

```
% 定义函数
t=linspace(0,22*pi,1100);
r=exp(cos(t))-2*cos(4*t)+sin(t./12)).^5;
subplot(2,1,1)
p=polar(t,r);             % 在极坐标系中绘图
subplot(2,1,2)
[x,y]=pol2cart(t,r);      % 找到笛卡儿坐标
```

```
plot(x,y)                  %  在x-y平面上绘图
```

可以得到两个不同的图形如图 13-13所示(上图)和(下图)。

```
[x,y] = pol2cart(t,r);  % Find the cartesian coordinates.

plot(x,y);              % Plot in x-y plane.
```

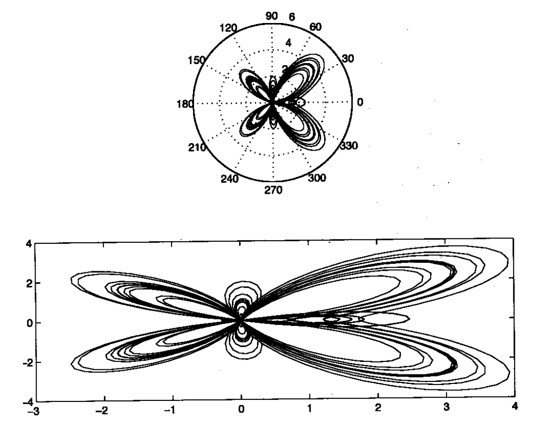


图13-13　在极坐标系中的图形(上图)和笛卡儿坐标系中的图形(下图)

■

可以使用quiver、feather、compass以及rose命令来绘制复数图形。这些命令同样可以用于实数矩阵；参见例 13.16(b)。

命令集129　复平面图形

quiver(X,Y)	对每对坐标(X_{ij}, Y_{ij})绘制一个箭头。命令(real(Z),imag(Z))可以看成绘制矩阵Z中复数元素方向与大小的图形。
quiver(x,y,dx,dy)	在坐标(x_i, y_i)处画一个箭头，由(dx_i, dy_i)指定方向和大小。
quiver(X,Y,Dx,Dy)	在坐标(X_{ij}, Y_{ij})处画一个箭头，由(DX_{ij}, Dy_{ij})指定方向和大小。
quiver(X,Y,···,s)	同上所述绘制箭头，但是用s进行标记。如果省略s，缺省值为s=1。
quiver(X,Y,···,str)	使用**str**指定的线型绘制箭头；参见表 13-1。
feather(Z)	把复数矩阵 **Z**中元素的相角和幅值显示成沿横轴等间隔辐射的箭头。
feather(X,Y)	等价于feather(X+Y*i)。
feather(Z,str)	使用**str**确定的线型绘制箭头;参见表13-1。
compass(Z)	把复数矩阵Z中元素的相角和幅值显示成从原点辐射的箭头。
compass(X,Y)	等价于compass(X+Y*i)。
compass(Z,str)	使用**str**确定的线型绘制箭头;参见表13-1。
rose(v)	绘制相角直方图，也就是向量 **v**中相角频率的角度直方图，间隔数为36。
rose(v,n)	同上，但是由*n*指定间距数。
rose(v,x)	绘制相角直方图，使用向量 **x**确定的间隔。

调色板部分的图P-5用四条极值曲线展示了对于函数 $z=f(x, y)$的箭图。

■ **例**13.7

定义Z：

$$\mathbf{Z}=\begin{pmatrix} 1+i & 2-i & 3-5i \\ -4+3i & 5-3i & i \\ -1-i & 3+3i & -1 \end{pmatrix}$$

下列命令产生图13-14的结果。

```
clear i;  % 确保i为复数。
Z = [1+i 2-i 3-5i; -4+3i 5-3i i; -1-i 3+3i -1];

clf;
subplot(2,2,1);  quiver(real(Z),imag(Z)); title('quiver');
subplot(2,2,2);  feather(Z);              title('feather')
subplot(2,2,3);  compass(Z);              title('compass')
subplot(2,2,4);  rose(angle(Z(:)));       title('rose');
```

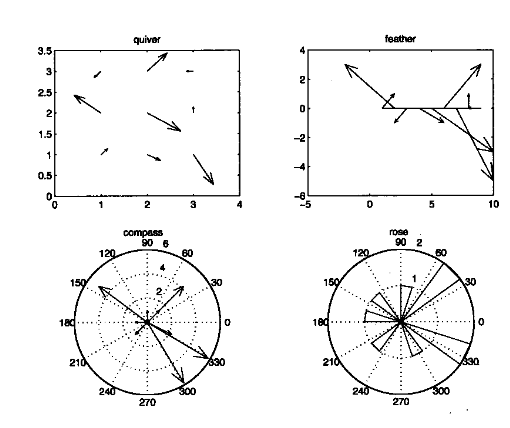


图13-14 图形表示的复数

使用subplot命令可以在一个图形窗口中绘制多个图形，参见 13-3节。这些命令给出如图13-14所示的图形。

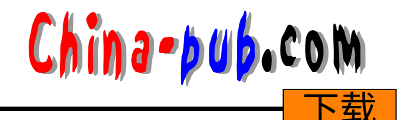
■

13.3 图形控制

MATLAB中的图形是面向对象的。本章中列出的命令通常用来设定和创建对象，这些对象用来绘制和改变图形。本节将讨论几个用来改变这些对象的高级命令，其中，许多改变是通过单一的命令来实现的。当然，还可以对单一图形对象的属性进行操作，这些将在第 14章中介绍高级绘图命令时进行详细介绍。

MATLAB中有两类窗口：命令窗口和图形窗口。给出 MATLAB命令要使用命令窗口，而图形窗口是用来展示图形的。图形窗口可以用本节所介绍的命令进行控制，也可以用另一种更直接的方式进行控制；参见第14章。

能否在屏幕上同时显示图形窗口和命令窗口由硬件来决定。但是有些命令可以用来在各



个窗口之间进行切换、清除窗口内容或保持当前图形。

命令集130 窗口命令

`figure(gcf)`	显示当前图形窗口。figure命令还可以用来在两个图形窗口之间进行切换和创建新的图形窗口；参见14.2节。
`shg`	显示当前图形窗口，等价于figure(gcf)。
`clf`	清除当前图形窗口。警告：如果设置hold on状态，窗口内容也将被清除。
`clg`	早期版本中等价于clf命令。在MATLAB以后的版本中可能会被淘汰。
`clc`	清除命令窗口。
`home`	移动光标到命令窗口的左上角。
`hold on`	保持当前图形。允许在当前图形状态下，使用同样的缩放比例加入另一个图形。
`hold off`	释放图形窗口，这样下一个图形将称为当前图形。这是缺省状态。
`hold`	在hold on和hold off之间进行切换。
`ishold`	如果当前图形处于hold on状态，则返回1；否则，返回0。

命令subplot用于在同一个图形窗口中绘制几个图形。subplot本身并不绘制任何图形，但是，它决定了如何分割图形窗口以及下一幅图将被画在哪个子窗口中。

命令集131 子图

`subplot(m,n,p)`	将图形窗口分割成m行n列,并设置p所指定的子窗口为当前窗口。子窗口按行由左至右，由上至下进行编号。这一命令在MATLAB的当前版本中也被写作subplot(mnp)。
`subplot`	设置图形窗口为缺省模式，即单窗口模式。等价于subplot(1,1,1)。

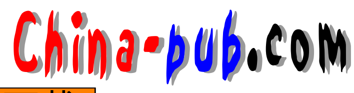
■ **例13.8**

(a) 在命令窗口的左上角显示一个由变化的随机数组成的矩阵。

```
clc                   % 清除命令窗口的内容
for I=1:10
    home              % 移动光标至左上角
    A=rand(5)         % 创建并输出矩阵
    pause(1);         % 延迟一秒钟
end
```

(b) 下列的MATLAB命令在左上角的子窗口中绘制出函数$f(x)=-x\sin x$的图形，在右上角的子窗口中绘制其导函数$f'(x)=-x\cos x-\sin x$的图形，在左下角的子窗口中绘制近似导函数的图形，在右下角的子窗口中绘制精确导函数和近似导函数的相对误差图形。

```
% 创建x的值。生成f的值y11,导函数y12,近似导函数y21和它的误差y22。
n = 1000;
x = linspace(-10,10,n);
```




```
y11 = (-x).*sin(x);
y12 = (-x).*cos(x) - sin(x);
y21 = diff(y11)./(x(2)-x(1));
y22 = (y21 - y12(1:n-1));
% 在左上角绘图
subplot(2,2,1); plot(x,y11);
title('The function')
% 在右上角绘图
subplot(2,2,2); plot(x,y12);
title('The derivative')
% 在左下角绘图
subplot(2,2,3); plot(x(1:n-1),y21);
title('The approximated derivative')
% 在右下角绘图
subplot(2,2,4); plot(x(1:n-1),y22);
title('The difference')
```

上述命令给出如图13-15的图形。

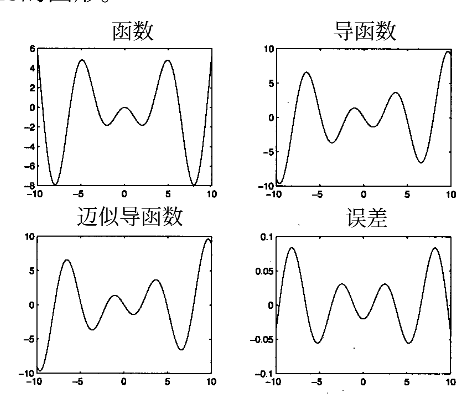


图13-15 函数$x$sin$x$及其导函数、近似导函数和误差图形

■

命令subplot也可用于三维图形，并且子图在同一窗口中可以有不同的大小，如下面的例子所示。

■ 例13.9

本例包含的M文件用来计算Mandelbrot不规则碎片，并将其用三种方式显示出来。该程序在由用户定义在复平面的网格中画点，并且根据下面的算法迭代所定义的网格中的每个数字 c。

$$z_0 = 0$$

$$z_{i+1} = z_i^2 + c$$

如果z_i是发散的，则当前c将不作为Mandelbrot集合的一部分。复平面上每一点 c的迭代次数将以与网格中相同的大小保存在矩阵 Mandelbrot中。算法将只迭代100次，因此分散点在Mandelbrot矩阵中的值为100。

```
% Mandelbro程序MandelbrotProg.m。
```

```
clear;
renum=input('renum:');        % 读实数点的个数
imnum=input('imnum:');        % 读虚数点的个数
remin=-2;remax=1;             % 定义要计算的数字
immin=-1.5;immax=1.5;
% 大小合适的向量

reval1 = linspace(remin,remax,renum);
imval1 = linspace(immin,immax,imnum);

% 虚平面上的网格
[Reval, Imval] = meshgrid(reval1,imval1);

Imvalreal = Imval; Imval = Imval*i;
Cgrid = Reval + Imval;

for reind = 1:renum
  disp(['reind = ',int2str(reind)]);        % 写循环状态

  % 迭代循环          z(i+1) = (z(i))^2 + c.
  for imind = 1:imnum
    c = Cgrid(reind,imind);
    numc = 0;                               % 初始化
    zold = 0.0 + i*0.0;
    z    = zold^2 + c;                      % z(0) = c.

    while (abs(z) <= 2) & ...
          (numc < 100)
      numc = numc + 1;
      zold = z;
      z = zold^2 + c;                       % 新z!
    end

    % 在矩阵mandelbrot中mandelbrot(n,m)位置上写入点
    Cgrid(n, m)的迭代次数

    Mandelbrot(reind,imind) = numc;
  end
end

% 用三种方式显示mandelbrot
矩阵图形：

clf;                                    % 清除图形

subplot(2,2,1);                         % 左上角
mesh(reval1,imval1,Mandelbrot);         % 绘三维网格表面图
axis([-2 1 -1.5 1.5 0 100])             % 改变坐标轴区间

subplot(2,2,2);                         % 右上角
contour(reval1,imval1,Mandelbrot,100);  % 等高线图
```

```
grid;                                 %  添加网格

subplot(2,1,2);                       %  降低图的位置(仅一个)
surf(Reval,Imvalreal,Mandelbrot);     %  Mandelbrot的表面图

view(2);                              %  俯视图

%  每个元素只有一种颜色，并且使用
%  反白的黑色条
shading flat;
colormap(flipud(jet));

%  显示颜色条。改变坐标轴
colorbar; axis([-2 1 -1.5 1.5]);
```

该程序的结果如图13-16所示。

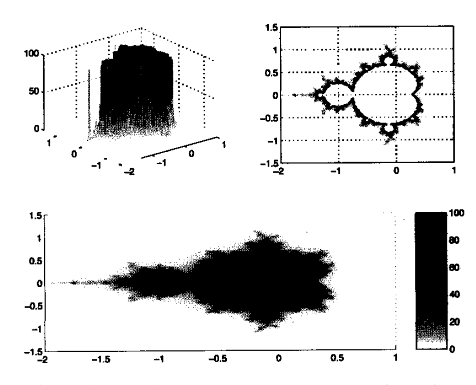


图13-16　以三种方式显示的Mandelbrot fractal:左上角：三维表面图；右上角：等高线图；下方：俯视图

■

图形几乎可以被放置在图形窗口的任何位置，还有许多的图形控制；参见第 14章。

每幅图的坐标轴通常都自动标上适合窗口中所有点的缩放比例。因此图的各个角就被定义成如下的形式：

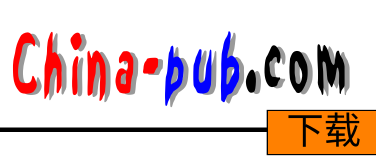
$$(\min(\mathbf{x}), \min(\mathbf{y})), (\max(\mathbf{x}), \min(\mathbf{y})), (\min(\mathbf{x}), \max(\mathbf{y})), (\max(\mathbf{x}), \max(\mathbf{y})).$$

有时，一些点由于缩放比例原因会与坐标轴重合，因此看到这些点比较困难。幸运的是，MATLAB 中的命令axis可以用来改变缩放比例。

同样可以使用鼠标或zoom命令来改变缩放比例。

命令集132　坐标轴，刻度和窗体缩放

axis	用行向量中给出的值，设置坐标轴的最大和最小值。对于二维图形，该向量中含有元素：[x_{min}, x_{max}, y_{min}, y_{max}]。对于三维图形(见13.5节)，是[x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}]。
axis(str)	字符串str的不同将给出不同的结果： ‘manual’固定坐标轴刻度。如果当前图形窗口为hold打开状态，则后面的图形将采用同样的刻度。



	‘auto’	把坐标轴刻度重新设置为缺省状态值。
	‘equal’	设置x轴和y轴为同样的刻度增量。
	‘tight’	采用与x轴方向和y轴方向相同比例的坐标轴，从而只绘制包含数据的部分坐标。
	‘fill’	设定坐标轴的边界，以使其能够匹配数据集的范围。
	‘ij’	翻转y轴，使得正数在下，负数在上。
	‘xy’	复位y轴，使正数在上。
	‘image’	重新设置图形窗口的大小，使得各像素有与宽度相同的高度以适应于计算机。
	‘square’	重新定义图形窗口的大小，使窗口为正方形。
	‘vis3d’	锁定坐标轴之间的关系。比如用在旋转3D对象时。
	‘normal’	复位图形窗口至标准大小。
	‘off’	不显示坐标轴或刻度。
	‘on’	显示坐标轴和刻度。

这一命令也可以写成axis normal等的形式。

`axis(v)` 根据向量v设置坐标轴刻度，使$x_{min}=v_1$, $x_{max}=v_2$, $y_{min}=v_3$, $y_{max}=v_4$。对于三维图形还会设置$z_{min}=v_5$, $z_{max}=v_6$。对于本章前面所讨论的对数图形，使用原数值，而不是对数值。通常还可以使用axlimdlg命令设置缩放比例；参见命令集169。

`axis(axis)` 固定坐标轴刻度。使得MATLAB在向原图上增加图形时不能改变刻度；参见13.3节命令hold。

`xlim([xmin xmax])` 设置$x_{min}=xmin$, $x_{max}=xmax$。

`xlim` 返回[x_{min}, x_{max}]。

`ylim([ymin ymax])` 设置$y_{min}=ymin$, $y_{max}=ymax$。

`ylim` 返回[y_{min}, y_{max}]。

`zlim([zmin zmax])` 设置$z_{min}=zmin$, $z_{max}=zmax$。

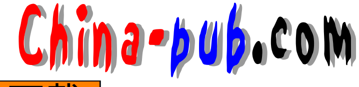
`zlim` 返回[z_{min}, z_{max}]。

`box` 控制是否将图形用坐标轴从各个边包围。命令box on打开该功能,而box off关闭该功能。只改变box就能在这两种状态之间进行切换。这一命令也同样适用于3D图形。

`datetick(axis, format)` 根据日期格式format格式化在坐标轴axis上的文本。参数**axis**可以是‘**x**’(缺省值)，‘**y**’ 或者‘**z**’。可参见2.5节可获得关于日期格式的更多信息。

`dragrect(X,step)` 允许用户在屏幕上拖动矩形。这些矩形是由n×4的矩阵X中每一行确定的。如果给定*step*，则只能在所给大小的偶数次内拖动矩形。

`grid on` 在图形窗口中画出网格。如果前面的图形是比如用极坐标(参见13.2节)绘制的，则网格也将采用极坐标绘制。



grid off	从图形窗口中清除网格。
grid	在grid on和grid off之间切换。
zoom on	使得用户可以在图形窗口中通过点击鼠标左键来放大二维图形，点击右键就缩小二维图形。还可以通过“点击和拖动”来选定一个区域。调整坐标轴刻度使得选中的区域占满整个图形窗口。
zoom off	关闭zoom功能。
zoom out	复位为满刻度。
zoom	在zoom on和zoom off之间切换。
zoom(factor)	用*factor*缩放当前坐标轴。
zoom axis	标定坐标轴。如果*axis*是xon或yon，则仅标定坐标轴。

关于坐标轴和网格的控制更详细的内容可参考第 14章。与axis命令关系最为密切的是caxis和saxis命令，分别用来设定颜色和声音的比例；参见13.6节和13.8节。

正如2.3节所描述的那样，命令可以看作是带有字符串参数的函数。因此，axis('square')也可以写成axis square，而grid off 也等价于grid('off')。

■ **例**13.10

(a) 定义一个值的集合来表示单位圆：

```
t=0:0.2:2*pi+0.2;        % 角度参数。
x=sin(t);                % x的值。
y=cos(t);                % y的值。
```

下面的命令给出如图13-17所示的图形：

```
plot(x, y,'—');
```

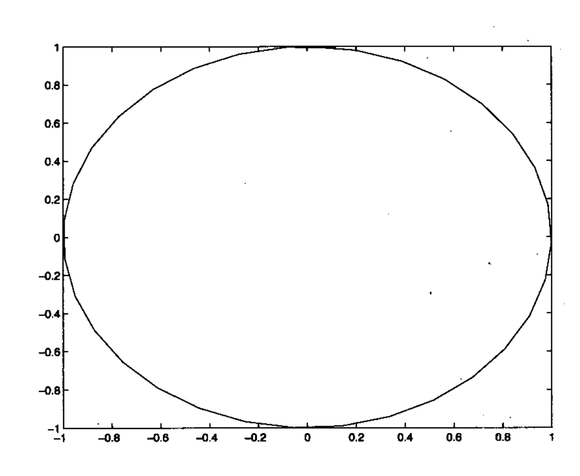
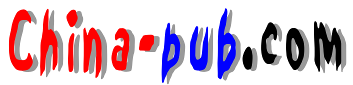


图13-17 如果比例不调整，圆看上去就象椭圆

(b)可以重新定义坐标轴的刻度使得圆看上去象圆，并且这次画上网格。

```
axis('square');         % 调整刻度
grid on;                % 绘制网格
```

这些命令可以得到图13-18上半部分的图形。下半部的图形是通过下列命令得到的：




```
axis('normal');              % 坐标轴复位
grid off;                    % 关闭网格
axis([-2 2 -3 3]);           % 改变坐标轴刻度
```

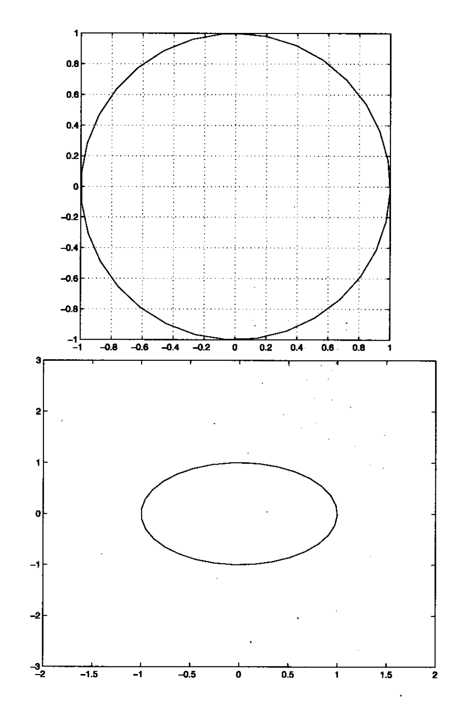


图13-18　上图:坐标轴设为正方形；下图:手动设定坐标轴刻度

■

有若干个命令可以用来在图形窗口中输出文本。命令 title、xlabel、ylabel和zlabel都可以输出标准文本项。MATLAB拥有大量的特殊字符和一些格式化函数，这些内容都在MATLAB Help Desk中关于文本部分做了详细的介绍。所有这些命令都可以用在当前子窗口中，并且通常应该在窗口中画完图形之后给出。要改变字体和其他属性，参见 14.2节。

命令集133　图形窗口中的文本

命令	说明
`title(txt)`	在图形窗口顶端的中间位置输出字符串 **txt**作为标题。
`xlabel(txt)`	在*x*轴下的中间位置输出字符串 **txt**作为标注。
`ylabel(txt)`	在*y*轴边上的中间位置输出字符串 **txt**作为标注。
`zlabel(txt)`	在*z*轴边上的中间位置输出字符串 **txt**作为标注。
`text(x,y,txt)`	在图形窗口的(*x*, *y*)处写字符串**txt**。坐标*x*和*y*按照与所绘制图形相同的刻度给出。对于向量**x**和**y**，字符串**txt**写在(x_i, y_i)的位置上。如果**txt**是一个字符串向量，即一个字符矩阵，且与*x*, *y*有相同的行数，则第*i*行的字符串将写在图形窗口的(x_i, y_i)的位置上。
`text(x,y,txt,'sc')`	在图形窗口的 (*x*, *y*)处输出字符串 **txt**，给定左下角的坐标为(0.0, 0.0),右上角的坐标则为(1.0, 1.0)。
`gtext(txt)`	通过使用鼠标或方向键，移动图形窗口中的十字光标，

	让用户将字符串**txt**放置在图形窗口中。当十字光标走到所期望的位置时，用户按下任意键或鼠标上的任意按钮，字符串将会写入在窗口中。
`legend(str1,str2,...pos)`	在当前图上输出图例，并用说明性字符串**str1**, **str2**等作为标注。如果指定参数*pos*，则图例将按下面所述放置： −1：　将图例框放在坐标轴外的右侧。 0：　将图例框放在坐标轴内侧，以便最少的点被覆盖。 1：　将图例框放在右上角。 2：　将图例框放在左上角。 3：　将图例框放在左下角。 4：　将图例框放在右下角。 [*x*, *y*]　将图例框的左下角移动到坐标(*x*, *y*)指定的位置。
`legend(H, str1, str2,...)`	返回句柄**H**以得到适当曲线的合适字符串。这对于用矩阵作为输入来画图是必要的。
`legend off`	从当前图形中清除图例。

命令`num2str`、`int2str`、`sprintf`等用于将数字转化成字符串的命令(参见5.1.2节)是十分有用的，甚至有时还可以和文本命令一起使用。

■ **例**13.11

(a)下面这个简单程序是用来完成随机路径的，可以看作是模拟空气粒子的运动。

该程序保存为particle.m：

```
% 随机路径。一个粒子从原点出发，随机地向任意方向移动，每步移动半个单位。
disp('Give the number of steps')          % 步数。
n=input('>>> ');
x=cumsum(rand(n, 1)-0.5);                 % 随机x值。
y=cumsum(rand(n, 1)-0.5);                 % 随机y值。
clf;                                      % 清除图形窗口。
plot(x,y);                                % 绘制路径。
hold on;                                  % 保持当前图形。
plot(x(1), y(1),'o', x(n), y(n),'o');     % 标记起点/终点。
axs=axis;                                 % 获取最小值和最大值。
scale=axs(2)-axs(1);                      % 计算比例。
text(x(1)+scale/30, y(1),'Start');        % 在起点和终点右侧。
text(x(n)+scale/30, y(n),'Finish');       % 标注文本。
hold off;                                 % 将保持状态开关恢复到标准状态。
xlabel('x'); ylabel('y');
title('Random walk')
```

键入命令`particle`来运行程序，将有：

```
Give the number of steps
>>> 100
```

现在，得到图13-19。



(b) 如果对程序中写文本的几行命令加以改动，比如，改为：

```
text(x(1)+scale/30,y(1),'Start','FontSize',20,'FontName',...
                                                'Times');
text(x(n)+scale/30,y(n),'Finish','FontSize',20,'FontName',...
                                                'Times');
xlabel('x','FontSize',18,'FontWeight','bold');
ylabel('y','FontSize',18,'FontWeight','bold');
title('Random walk','FontSize',18,'FontWeight','bold');
```

得到图13-20：

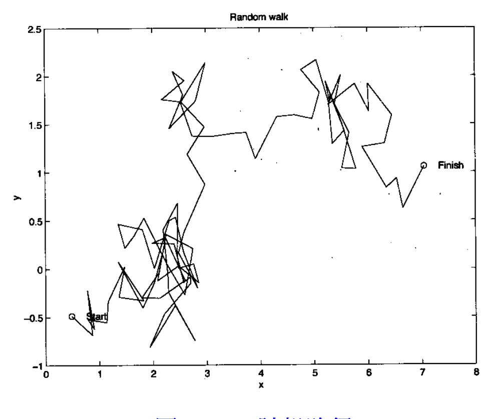


图13-19 随机路径

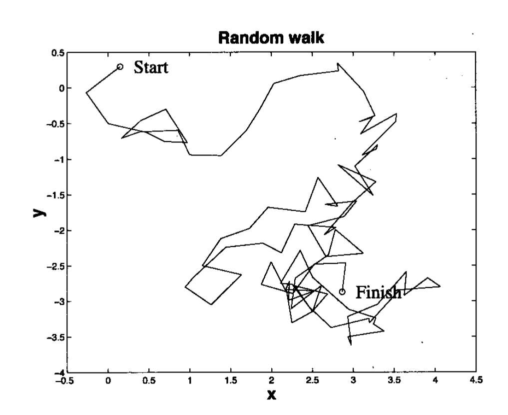


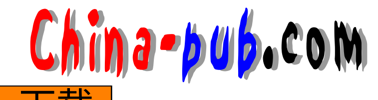
图13-20 改变文本格式后的随机路径

■

命令ginput通常用于从图形窗口中获取信息。该命令在图形窗口中放置一个光标，用户可以通过键盘或鼠标移动此光标。当移动到预定位置时，按下任意键或鼠标按钮，坐标值将被传递到MATLAB中。如果没有指定读取的坐标数目，MATLAB将读取每一个键盘或鼠标按钮按下时的值，直到按‘回车’键停止。

命令集134 从图形窗口中读取数据

`[x,y]=ginput`	从图形窗口中读取坐标值。在图形窗口中放置一个光标，用户可以通过鼠标或方向键对光标进行定位，并且通过按下鼠标按钮或键盘上任意键，将坐标值传递到 MATLAB中。这些坐标值保存在向量x和y中。这一过程直到按下‘回车’键才终止。
`[x,y]=ginput(n)`	从图形窗口中读取n个坐标值。
`[x,y,t]=ginput(···)`	按下鼠标键以整数值的形式返回到向量t中。按下左键返回1，中键返回2，右键返回3。如果使用键盘，将返回按下键的ASCII码。
`waitforbuttonpress`	停止运行MATLAB直到在当前图中按下鼠标或键盘上的任意键。如果按下鼠标，则waitforbuttonpress返回0；如果按下键盘上的任意键，则返回1。
`rbbox`	在当前图的橡皮圈框周围画虚线。可以和waitforbuttonpress一起使用来进行动态控制。比如：在zoom中使用该命令。



■ **例**13.12

在MATLAB4中可以使用ginput命令重新定义图形窗口，以使得(0，0)为左下角而(1，1)为右上角。而在MATLAB现在的版本中，不再这样使用。但是，使用下面用户所定义的函数ginput01可以很容易地创建一个这样的窗口。

```
function [x,y,button] = ginput01(N);

if (nargin == 0), N = inf; end

[x,y,button] = ginput(N);
xylim = get(gca,{'xlim','ylim'}); % 获取坐标轴的范围
x = (x-xylim{1}(1))/diff(xylim{1});
y = (y-xylim{2}(1))/diff(xylim{2}); % 用坐标轴范围重新定义X,Y 的大小
```

■

get命令在将14章中介绍。

■ **例**13.13

ginput和waitforbuttonpress命令提供给MATLAB程序员用来建立简单的交互式程序。下面的M文件就使用这两个命令来绘制由用户确定的点连成图形。绘制完成后，程序等待，当用户点击图形时，删除图形。

```
% 交互式绘图的M文件

n = figure;                    % 创建新的图形

disp('To draw a line in the figure:')
disp('Press the left mouse button in the figure for start,')
disp('each bend and stop of the line. ')
disp('Press right mouse button when finished.')

[x,y,t] = ginput(1);           % 读取第一次鼠标键按下时的坐标
plot(x,y,'o')                  % 用圆圈作一标记
xx = x; yy = y;                % 保存坐标
hold on; axis([0 1 0 1]);      % 保持图形锁定坐标轴

while t ~= 3                   % 如果不是右键按下
  [x,y,t] = ginput(1);         % 则读新的坐标
  plot(x,y,'o')                % 并用圆圈作一标记
  xx = [xx x];                 % 保存坐标
  yy = [yy y];
end

clf; line(xx,yy);              % 清除图形并画一条线

disp('Click on the figure when you are done')

waitforbuttonpress;            % 等待直到用户在图形中按下鼠标注意键
delete(n);                     % 清除图形
```

命令figure、delete和line将在14.2节中介绍。其他的交互式命令在 14.3节中介绍。



上面的M文件运行后给出如图13-21所示的图形。

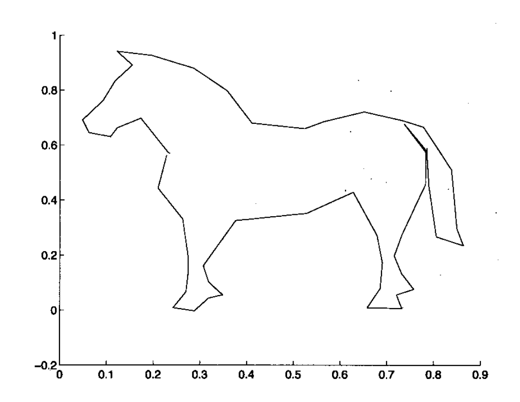


图13-21 例13.13中交互式程序运行的结果

■

13.4 生成网格和绘制等高线图

可以在二维和三维空间中画出带有两个变量的函数如z=f(*x*, *y*)的等高线图形。前者用contour命令完成，而后者可用命令contour3完成。值得注意的是，MATLAB5可以处理不统一的网格。

命令集135 等高线图形

contour(Z)	绘制矩阵**Z**的等高线图形。将**Z**中各元素看成是高于(*x*, *y*)平面的高度。如果**Z**是一个*m*×*n*的矩阵，则横轴的刻度设为1到*n*，而纵轴的刻度设为1到*m*。命令c=coutour(Z)返回将被如命令clabel使用的等高线矩阵C，参见contourc。
contour(Z,n)	绘制*n*条等高线。如果不指定*n*，则绘制10条。
contour(Z,v)	绘制向量**v**中的值所确定高度的等高线。
contour(x,y,Z)	用向量**x**和**y**作为矩阵**Z**的坐标绘制**Z**的等高线图形，即用**x**和**y**设置坐标轴的刻度。
contour(x,y,Z,n)	用**x**和**y**设置坐标轴的刻度，绘制*n*条等高线。
contour(x,y,Z,v)	用**x**和**y**设置坐标轴的刻度，绘制向量v中的值所确定的高度的等高线。
contour(···,str)	用字符串**str**指定的线型和颜色绘制等高线。参见13.1节的plot指令和表13-1。
C=contourc(···)	不画出等高线，而使用contour和clabel计算出等高线矩阵**C**。**C**是一个两行矩阵，对于每条等高曲线都是连续保存它们的图段。使用type help contour 可以获得更多的信息。
C=contours(···)	计算等高线矩阵**C**。当边界为非矩形区域时，contour使用该矩阵。type help contour可以获得更多的信息。
contourf(Z)	绘制矩阵**Z**的填充等高线。与contour使用相同的参数。
contour3(x,y,z,n)	绘制*n*条三维等高线，即不将等高线投影到(*x*, *y*)平面上。返回clabel使用的等高线矩阵。

clabel(C)	在等高线图形上增加高度标记。标记的位置是任意选择的。矩阵C是用命令contour和contourc返回的等高线矩阵。
clabel(C,v)	用向量v中给出的等高线水平进行标记。矩阵 C是用命令contour和contourc返回的等高线矩阵。
clabel(C,'manual')	在鼠标指定的位置放置等高标记。用户可以通过鼠标或键盘上的方向键来移动等高线图形中的光标，当按下某键时，写入数字进行标记。按‘ 回车 ’终止整个操作。

如果要画出已定义矩阵Z的等高线图形，可以使用contour(Z)或contour3(Z)命令。调色板一节的图P-4用四条极值曲线展示了对函数$z=f(x, y)$使用contourf命令的结果。

■ **例**13.14

(a) 假定已定义了图13-35中二维函数表面图的矩阵Z。那么，下列命令将得到图13-22所示的图形：

```
[X,Y] = meshgrid(-3:1/8:3);
Z     = peaks(X,Y).*sin(X);

v1 = -4:-1;
v2 = 0:4;

clf;

subplot(2,1,1);          % 上方子图
contour(Z,v1,'k-');      % 对Z的负数绘制实线
hold on;

contour(Z,v2,'k--');     % 对Z的非负数绘制实线
hold off;

subplot(2,1,2);          % 下方子图
C = contour(Z);          % 绘制等高线
clabel(C);               % 加入等高标记
```

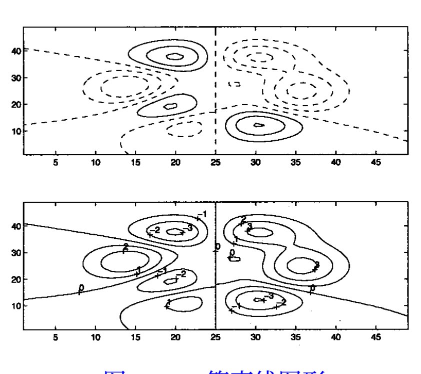


图13-22　等高线图形

(b) 也可以将contour用于非矩形网格。运行例 13.23中的程序可以得到图 13-23。在图13-23中重新设置了文本字体来说明每个等高线可以有自己的图形句柄。参见第 14章中关于图



形对象和句柄部分以获得更多的信息。

contourf命令填充线与线之间的区域；如图 13-24所示。该图是用白线代替黑线来绘制的；参见例13.23。

尽管如此，计算Z还是很有必要的。这可以用两步来完成。首先，在希望绘制等高线的区域定义一个网格。该区域由长度分别为 *n*和*m*的向量**x**和**y**来定义，这样 *x*和*y*的值均在网格中。

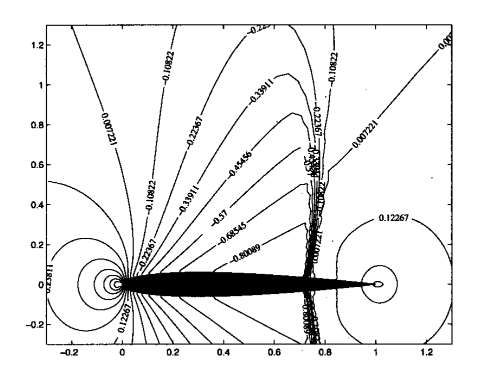


图13-23　气压的等高线、等压线和15个高度水平的剖面图

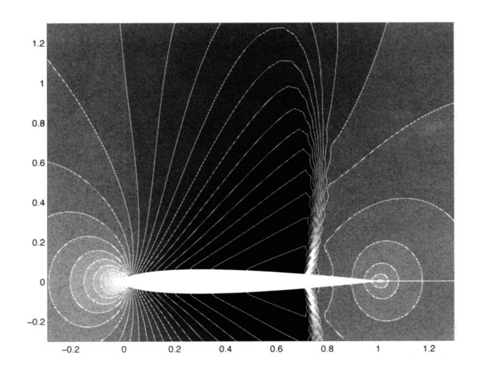


图13-24　contourf创建的等高线剖面图。高度水平数为30，且均匀分布在－1～1之间

■

注意：这里**x**和**y**的元素不一定是等距的。然后，用命令[U,V]=meshgrid(x,y)来形成网格。实际上，网格就是两个矩阵**U**和**V**，包含着它的*x*和*y*坐标。矩阵**U**由复制*m*行的向量**x**组成，而**V**由复制*n*列的向量**y**组成。如图13-25所示，*y*轴方向向下强调网格点和矩阵元素之间的一致性。

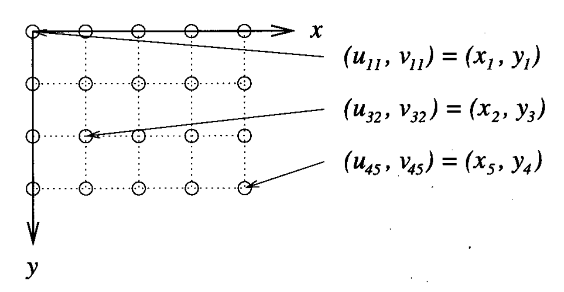


图13-25　**x**的五个值和**y**的四个值形成的网格定义两个4×5的矩阵**U**和**V**，矩阵的元素来自向量**x**和**y**

使用meshgrid命令同样可以生成柱形网格和球形网格。

命令集136　网格的生成

[U,V]=meshgrid(x,y)	用来自向量**x**和**y**的*x*坐标和*y*坐标形成网格，并生成矩阵。长度为*n*的向量**x**包含升序排列的*x*坐标，而长度为*m*的向量**y**包含升序排列的**y**坐标，分别复制*m*和*n*次形成两个*m*×*n*的矩阵**U**和**V**。这些矩阵表示整个矩形区域内的 *x*和*y*坐标。坐标对(u_{ij},v_{ij})，$i=1,\cdots,m$，$j=1,\cdots,n$，可通过使用命令Z=f(U,V)用来计算$z_{ij}=f(u_{ij}, v_{ij})$，参见图13-25。

[U, V]=meshgrid(x)	等价于[U,V]=meshgrid(x,x)。
[U,V,W]= meshgrid(x,y,z)	以同样的方式产生三维网格，可用来对含有三个变量的函数进行求值。
[X,Y,Z]=cylinder (r,*n*)	象meshgrid一样返回坐标矩阵，返回的坐标形成圆柱体或圆锥体表面。圆柱体半径来自向量r，包含沿圆柱周围*n*个等距离的点。如果*n*不确定，缺省值为20。如果r和*n*都不确定，则令**r**=(1　1)，*n*=20。
cylinder(r,*n*)	同上画圆锥体，但不返回坐标。
[X,Y,Z]=sphere(n)	返回在矩阵**X**、**Y**和**Z**总共(*n*+1)×(*n*+1)个矩阵中的单位球体上的*n*个等距坐标。
sphere(n)	同上绘制球体图形，但不返回坐标。

在调色板一节所创建图P-2时多次用到sphere命令。

■ **例**13.15

假定在单位正方形上定义一个网格 **U**，**V**，并且要求沿*x*轴有5个网格点，沿*y*轴有4个网格点，如图13-25所示的那样。首先定义向量**x**和**y**，然后形成网格：

```
x = linspace(0,1,5);        % 定义x值
y = linspace(0,1,4);        % 定义y值

[U,V] = meshgrid(x,y)       % 形成网格
U =
          0    0.2500    0.5000    0.7500    1.0000
          0    0.2500    0.5000    0.7500    1.0000
          0    0.2500    0.5000    0.7500    1.0000
          0    0.2500    0.5000    0.7500    1.0000
V =
          0         0         0         0         0
     0.3333    0.3333    0.3333    0.3333    0.3333
     0.6667    0.6667    0.6667    0.6667    0.6667
     1.0000    1.0000    1.0000    1.0000    1.0000
```

■

第二步是在网格上对函数*z*=f(*x*, *y*)求值。在所定义的网格中，即Z=f(U,V)。这要求函数f用元素操作运算符来定义；参见3.5节。

■ **例**13.16

(a) 绘制下列三个函数的等高线图形。

$$\begin{cases} Z_1 = \mathbf{f}_1(x,y) = \sin x \cdot \sin y & x, y \in [0,\pi] \times [0,\pi] \\ Z_2 = \mathbf{f}_2(x,y) = x - 0.5x^3 + 0.2y^2 + 1 & x, y \in [-3,3] \times [-3,3] \\ Z_3 = \mathbf{f}_3(x,y) = \sin(\sqrt{x^2+y^2})/\sqrt{x^2+y^2} & x, y \in [-8,8] \times [-8,8] \end{cases}$$

程序的第一部分绘制网格并对函数求值。程序的最后部分绘制图形。这个程序为contours.m：

```
x = 0:0.2:3*pi;             % 生成坐标
y = 0:0.25:5*pi;
```

```
[XX,YY] = meshgrid(x,y);
Z1 = sin(XX).*sin(YY);            % 对Z1求值

x = -3:0.25:3;                    % 生成坐标
y = x;
[XX,YY] = meshgrid(x,y);
Z2 = XX - 0.5*XX.^3 + 0.2*YY.^2 + 1;   % 对Z2求值

x = -8:0.5:8;                     % 生成坐标
y = x;
[XX,YY] = meshgrid(x,y);
r = sqrt(XX.^2+YY.^2) + eps;
Z3 = sin(r)./r;                   % 对Z3求值

clf;

subplot(2,2,1); contour(Z1);
title('sin(x)*sin(y)');

subplot(2,2,2); contour(x,y,Z3);
title('sin(r)/r');
subplot(2,2,3); contour3(Z2,15);
title('x-0.5x^3 + 0.2y^2 + 1');

subplot(2,2,4); contour3(x,y,Z3);
title('sin(r)/r');

subplot(2,2,3); rotate3d;
```

最后一行程序允许用户通过使用鼠标对左下角的图形进行旋转以获得更好的视图；rotate3d将在13.5节中介绍。运行该程序，经过旋转后，将得到图 13-26所示的结果。

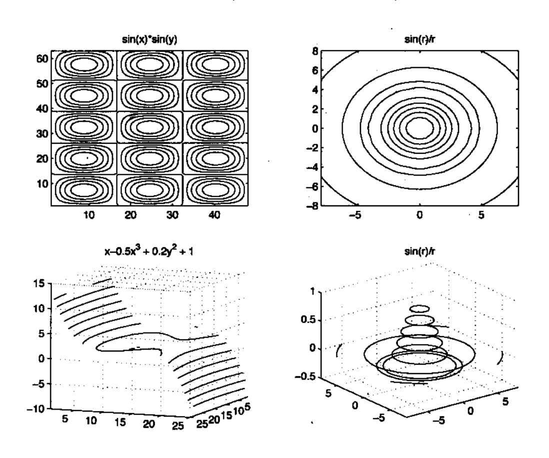


图13-26 一些函数的等高线图形

(b) 为了得到一个真实的函数侧面图，在等高线的图形中绘制梯度。梯度可以用命令gradient(参见6.2节)计算，并且可以用命令quiver绘图(参见13.2节)。这个有趣的图形可以通过下列语句得到：

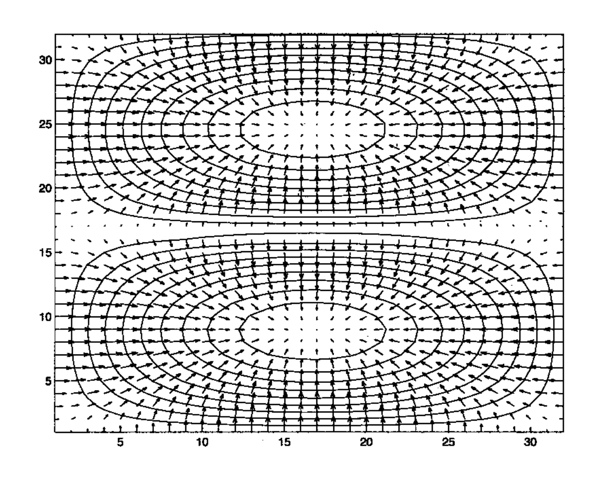


图13-27　带有梯度的等高线图形

```
[X,Y]       = meshgrid(-pi/2:0.1:pi/2,-pi:0.2:pi);
Z           = abs(sin(Y).*cos(X));
[DZDX,DZDY] = gradient(Z,.1,0.2);

contour(Z);         hold on;
quiver(DZDX,DZDY); hold off;
```

结果如图13-27所示。

■

13.5　三维图形

三维图形可以用命令plot3来绘制。该命令与plot类似，但是plot3需要3个向量或矩阵参数。与plot一样，线型和颜色可以用一个字符串来确定；参见表 13-1。

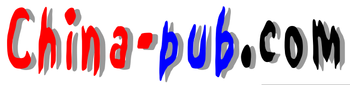
命令集137　三维图形

plot3(x,y,z)	用(x_i, y_i, z_i)所定义的点绘制图形。向量**x**、**y**和**z**必须为等长度的。
plot3(X,Y,Z)	对矩阵**X**、**Y**和**Z**的每一列绘图。这些矩阵必须大小相等。或者，也可以是长度与矩阵列向量相等的向量。
plot3(x,y,z,str)	使用字符串**str**确定的线型和颜色按照上面所述的方法绘制图形。参见表13-1以获得允许使用的字符串值。
plot3(x1,y1,z1, str1,x2,y2,z2, str2,···)	用字符串**str1**确定的线型和颜色对**x1,y1,z1**绘图，用字符串**str2**确定的线型和颜色对**x2, y2, z2**绘图…。如果省略**str1,str2,**…，MATLAB将自动选择线型和颜色。

■ **例**13.17

受例13.11的启发，现在可以编写一个程序来模拟三维空间的随机路径。程序 particle3.m如下：

```
% 三维空间的随机路径
% 用随机数创建向量X、Y和Z。绘制“路径”。
```




```
n = input('Give the number of steps : ');
x = cumsum(rand(1,n)-0.5);
y = cumsum(rand(1,n)-0.5);
z = cumsum(rand(1,n)-0.5);

plot3(x,y,z);                                    % 绘制路径
grid on;                                         % 显示网格
text(x(1),y(1),z(1),'Start','FontSize',20);      % 标记起点
text(x(n),y(n),z(n),'Finish','FontSize',20);     % 标记终点
```

运行程序，将得到如图13-28的结果。

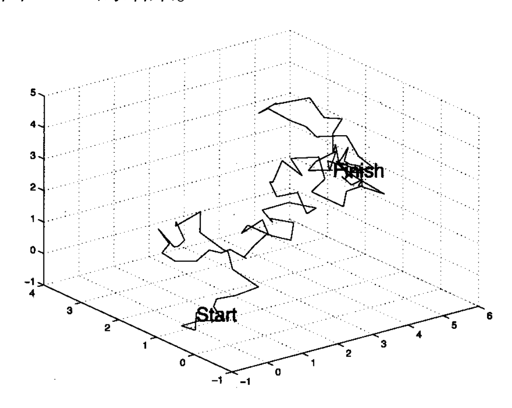


图13-28　三维空间中的随机路径

■

三维图形中的写文本命令与二维中相同，见title、text、xlabel、ylabel和zlabel，参见13.3节。

命令集138　三维绘图函数

bar3(x,A,*width*, *format*)	矩阵**A**的各列对向量**x**绘制竖直条形图。如果给定*width*，则每个竖直条的宽度为 *width*。如果给定字符串 **format**，format可以取下列字符串中的某个。 'detached' 给出默认的分离的饼图。 'grouped' 给出一组饼图。 'stacked' 给出堆叠式饼图。 linespec 使用指定的线条颜色；参见表13-1。
bar3(A)	等价于bar3(1:size(A,1),A)。
bar3h(x,A,*format*) *width*,	矩阵**A**的各列对向量x绘制水平条形图，如果给定*width*，则每个水平条的宽度为 *width*。如果给定字符串 **format**，**format**的取值与bar3相同；参见bar3所述。
pie3(x,*explode*)	类似饼图的三维饼图。
quiver3(x,y,z,u, v,w,s,*format*)	在由三个向量**x**、**y**和**z**所确定的坐标处绘制箭头。箭头的长度和方向由向量**u**、**v**和**w**确定，坐标刻度由*s*确定，*s*的默认值为1。字符串**format**是可选项，用来确定线条的格式。

`ribbon(x,y,`*`width`*`)`	向量**y**对**x**绘制三维丝带图，而不再是普通曲线。*width*指定丝带的宽度，缺省值为0.75。
`scatter3(x,y,z,` *`size,color`*`)`	在**x**, **y**和**z**所确定的点处绘制着色的圆圈图形，**x**，**y**和**z**的大小必须相等。可参考`scatter`以获得更多的信息。
`stem3(x,y,A)`	矩阵**A**对向量**x**和**y**绘制离散序列数据图。
`stem3(A)`	矩阵**A**对规格化的*xy*平面绘制离散序列数据图。
`stem3(···,format)`	按照**format**所指定的格式绘制离散序列数据图。**format**除了可以是标准线条格式外，还可以时字符串‘**filled**’，它表示将填充上方的圆圈。
`trimesh(Tri,x,y,z)`	按矩阵**Tri**绘制三角网格表面图。向量**x**，**y**，**z**定义三角形。参见第6章关于三角形的介绍。
`tirsurf(Tri,x,y,z)`	按矩阵**Tri**绘制三角区域的表面图。向量**x**，**y**和**z**定义三角形的各个角。

调色板一节的图P-6演示了命令`bar3`和`pie3`的使用。

■ **例**13.18

现有一向量`x=[1 2 3 4 5]`，要绘制一个三维饼形图，并且将其中最大的部分分离出来。为此，需要向量`explode=[0 0 0 0 1]`除与`x`中最大元素值相对应的元素为1外，其他元素均为0。如果输入`pie3(x,explode)`，将得到如图13-29所示的结果。

二维图形中，可以用`comet`命令动态显示。同样，三维图形中可以用`comet3`命令动态显示。

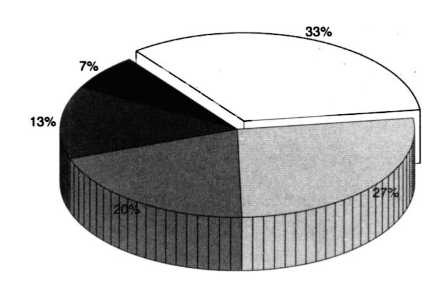


图13-29　最大部分分离出来的三维饼形图

■

命令集139　三维彗星图

`comet3(x,y,z)`	显示函数z=**f**(x, y)的一个三维彗星轨线动画，即一个带有位于(x_i, y_i, z_i)坐标之间的慧尾的星状图。
`comet3(x,y,z,`*`p`*`)`	显示同上所述的彗星轨线动画，慧长为p*length(y) p，的缺省值为0.1。

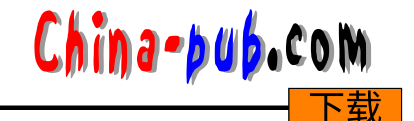
在MATLAB中，可以通过下面的方法画出函数z=**f**(x, y)的表面图：

1) 如13.4节所描述的那样形成网格；

2) 估计`Z=f(U,V)`，矩阵**U**和**V**分别是来自x和y的坐标形成的矩阵；

3) 用MATLAB中的某一表面图命令绘制表面图。注意：网格不一定是矩形。如果不是，则网格坐标必须包含在所调用的函数中。

命令集140中的命令用来绘制三维网格表面图。



命令集140 三维网格表面图

mesh(Z)	将矩阵**Z**中的各个元素作为矩形网格上的高度，对这些值绘图，并且将相邻的点连接形成三维网格表面图。颜色由高度，即 **Z** 中的元素指定。
mesh(Z,C)	将矩阵**Z**中的各个元素看成矩形网格上的高度，对这些值绘图。各点的颜色由矩阵**C**中各元素的值确定。
mesh(U,V,Z,C)	绘制矩阵**Z**中元素的函数网格表面图，相邻点用线连接。该图画在三维视图中，元素z_{ij}代表网格点(x_i, y_j)上的高度。观察点是自动设定的，可以使用view命令来更改。参数如下： **U** x坐标矩阵 **V** y坐标矩阵 **Z** z坐标的矩阵,通常z_{ij}=**f**(u_{ij}, v_{ij}) **C** 各点的颜色矩阵。如果**C**省略，则**C**=**Z**。
meshc(...)	与mesh一样绘制网格表面图，并在该图的下面绘制等高线图。
meshz(...)	与mesh一样绘制网格表面图，并在(x, y)平面上绘制参考平面。
waterfall(...)	类似meshz，但是参考平面只在一个方向上绘制。
hidden *val*	切换消隐线的移动状态。变量*val*可以为on，表示不绘制消隐线；也可以为off，表示绘制消隐线。这一命令只适用于命令mesh.

可以使用命令rot90对由矩阵定义的图形进行旋转。此外，在本节后面，还将看到 view 命令。使用rotate3d命令很容易决定一个观察点。

命令集141 矩阵旋转

rotate3d*val*	允许用户使用鼠标旋转一个三维图形。如果指定 *val*，则有两种可能，on表示打开该功能，或者off表示关闭该功能。
rot90(A)	返回逆时针旋转90度后的矩阵**A**。常常和命令mesh一起使用。
rot90(A,k)	返回逆时针旋转k×90度后的矩阵**A**。

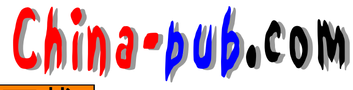
必须强调的是命令mesh对获得一个矩阵的图像非常有用。另外，它对理解数值方法也是十分有帮助的。

■ **例**13.19

(a)编写一个MATLAB程序来绘制与例13.16中相同函数的三维网格表面图。假定定义了例13.16中相同的矩阵**Z1**、**Z2**和**Z3**。给出下列语句：

```
% 矩阵Z1、Z2和Z3将在coutours中定义
%
% 绘制四个函数的图形

clear;
contoursProg;                    % 运行contours.m
```




```
disp('Matrices defined!');
clf;

subplot(2,2,1), mesh(Z1)
title('sin(x)*sin(y)');

subplot(2,2,2), meshz(Z2)
title('x - 0.5*x^3 + 0.2*y^2 + 1');

subplot(2,2,3), waterfall(Z2)
title('x - 0.5*x^3 + 0.2*y^2 + 1')

subplot(2,2,4), meshc(Z3)
title('sin(r)/r')
```

运行上面的程序，将得到如图13-30所示的结果。

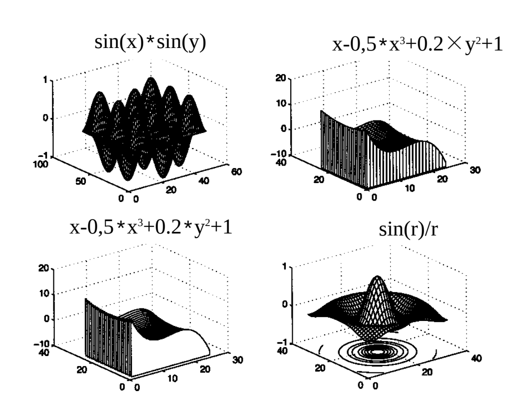


图13-30　用各种mesh 命令绘制的几个函数图形

(b) 下面的MATLAB程序用来对给定的矩阵**A**进行LU和QR分解，并在图形窗口中绘制矩阵**L**、**U**、**Q**和**R**的四个子图。这些命令保存在文件**luqrmesh.m**中：

```
if ~exist('A')
  A = input('Give a matrix A: ');
else
  disp('The following matrix exists:');
  A
end

[L,U] = lu(A);
[Q,R] = qr(A);
subplot; mesh(A); title('The matrix A');
disp('Press any key when you are ready!');
pause; clf;

subplot(221); mesh(L); title('The matrix L');
subplot(222); mesh(U); title('The matrix U');
subplot(223); mesh(Q); title('The matrix Q');
subplot(224); mesh(R); title('The matrix R');
```

假定已定义了矩阵A，输入命令luqrmesh运行程序：

```
The following matrix exists:
A =
    30     1     7     2     6     5
     4    24     9     4     0     1
     9     9    21     6     1     7
     8     3     7    21     3     9
     4     9     1     6    27     9
     1     6     4     2     3     9
Press any key when you are ready!
```

开始得到图13-31，按任意键后，将得到图13-32。

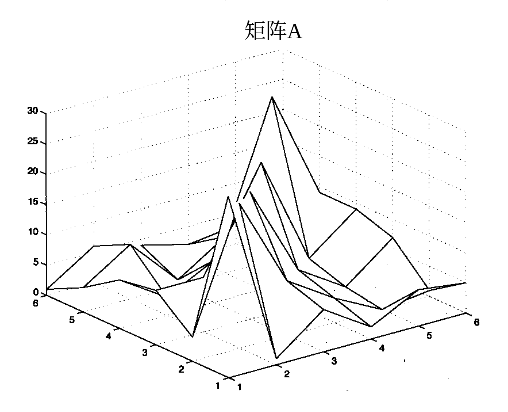


图13-31　矩阵A的三维网格表面图

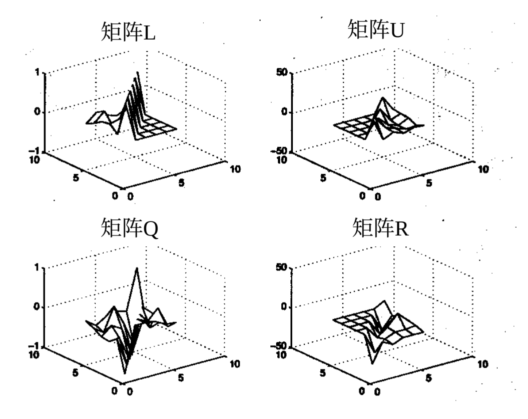


图13-32　矩阵L、U、Q和R的三维网格表面图

■

在MATLAB中有大量特殊的三维图形函数。

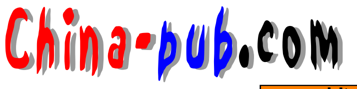
在本节开头详细讨论了如何绘制三维网格表面图。用同样的方式，还可以绘制阴影表面图：用各点的x和y坐标来创建两个矩阵，函数求值，用适当的命令绘制出图形。

用命令fill、fill3、surf、surfc和surfl也可以绘制出表面图形。命令surfl给出的带光照的阴影表面图。它是建立在漫反射、单向反射和周围的亮度模式的组合图基础之上的。这种表面图是由命令集146中定义的shading interp命令得到的，并且以灰度图的形式展现出来。使用这些数据可以创建出正常的表面向量，并且获得漫反射或单向反射的反射比。

命令集142　表面图和亮度

surf(X,Y,Z,C)	绘制出由坐标(X_{ij}, Y_{ij}, Z_{ij})确定的表面图形。如果 **X**和**Y**分别是长度为m和n的向量，那么，**Z**必须为$m\times n$的矩阵，并且表面是由(X_j, Y_i, Z_{ij})来定义的。如果省略参数 **X**和**Y**，MATLAB将使用统一的矩形网格。图形的颜色由矩阵 **C**中的元素定义。如果省略参数**C**，则缺省值为**C**=**Z**。
surfc(X,Y,Z,C)	除完成surf(···)所完成的功能外，还将在表面下方绘制等高线图。
surfl(X,Y,Z,ls)	与surf(···)功能类似，同时还在**ls**=[v, h]或**ls**=[X, Y, Z]方向上有一束光源入射，参数与命令view的参数相同。

surfl(X,Y,Z, ls,r)	与上面的命令类似，使用向量**r**=[*ambient*, *diffuse*, *specular*, *spread*]可以设置周围的亮度、漫反射、单向反射以及散射系数所产生的影响
surfnorm(X,Y,Z)	与surf相似，同时还画出法线。
[Nx,Ny,Nz]= surfnorm(X,Y,Z)	画出矩阵**X**、**Y**和**Z**定义的表面法线，但不绘制图形。因此，$(nx_{ij}, ny_{ij}, nz_{ij})$是定义点$(X_{ij}, Y_{ij}, Z_{ij})$处法线的向量。法线长度为1。
diffuse(Nx,Ny,Nz, ls)	使用兰伯特定律，返回法向量分量为 **Nx** ,**Ny**和**Nz**的表面漫反射率。**ls**是定义光源位置的三组分向量。
specular(Nx,Ny, Nz,ls,v)	使用光源位置**ls**和观察点**v**，返回法向量分量为**Nx**、**Ny**和**Nz**的表面反射率。
light	创建带有标准值的光源。
light(*propstr*, *val*,...)	创建光源，同时将它的属性**propstr**设置为*val*。可以同时对多个属性进行设置。设置属性需要一些关于图形对象的知识；参见第14章，特别是关于光源对象的表14-26。
lightangle (azimuth,*hight*)	创建无穷远处的光源对象，为了定位而使用球面系统，*azimuth*是水平的旋转角度，而*height*表示“高度”。
camlight (azimuth,*height*)	创建观察点坐标系统的光源。为了定位而使用球面系统，*azimuth*是水平的旋转角度，而*height*表示“高度”。
camlight headlight	在观察点位置创建光源。
camlight right	在观察点的右上方创建光源。
camlight left	在观察点的左上方创建光源。
camlight(···, type)	设置光源类型。字符串type可以是‘local’(缺省值)或‘infinite’。
lighting mode	改变多边形或表面对象，如用 surf或patch等绘制的图形亮度模式。模式值可以是：flat、gourand、phong和none。关于这些模式的更多的信息可在helpdesk中找到。
material mode	改变多边形或表面对象，如用 surf或patch等绘制的图形反射比模式。模式值可以是：skiny、dull和metal。更多的信息可参见helpdesk。
material({ka kd kn n sc})	为表面的反射比设置不同的值。参见helpdesk中关于不同属性的含义。
	ka　改变Ambientstrength的属性。
	kd　改DiffuseStrength的属性。
	kn　改变SpecularStrength的属性。
	n　改变SpecularExponent的属性。
	sc　改变SpecularColorReflectance的属性。
pcolor(Z)	绘制矩阵**Z**的伪色图为带有颜色的细胞矩形阵列。图形颜色



	由矩阵Z的元素值决定。
pcolor(X,Y,Z)	与surf(X,Y,Z)；view(2)相同；参见命令集143。
fill(x,y,c)	绘制出坐标向量 x和y确定边角的多边形。多边形用字符串 c 或与x，y长度相等的向量 c中的值指定的颜色 (参见表13-1)填充。如果x, y为矩阵，将对每一列绘制多边形。
fill3(x,y,z,c)	绘制x，y，z确定的多边形。绘制出坐标向量 x和y确定边角的多边形。多边形用字符串 c或向量c中的值指定的颜色 (参见表13-1)填充。如果参数为矩阵，将对每一列绘制多边形。

这些命令中使用的颜色刻度可以进行调整；参见 13.6节。

在调色板一节的图P-2中，使用了命令light来创建光源。

■ **例**13.20

(a) 绘制带有等高线的(sinr)/r函数图形。

```
x       = -8:0.5:8;
[Xx Yy] = meshgrid(x);

R = sqrt(Xx.^2+Yy.^2) + eps;
Z = sin(R)./R;
clf;
surfc(Xx,Yy,Z); title('(sin r)/r');
```

得到图13-33。

(b) 现在我们绘制同一个函数的带有法线的图形。假设 Xx、Yy和Z与(a)中一样已经计算出来。下面的语句将得到图13-34的结果。

```
surfnorm(Xx, Yy, Z)
```

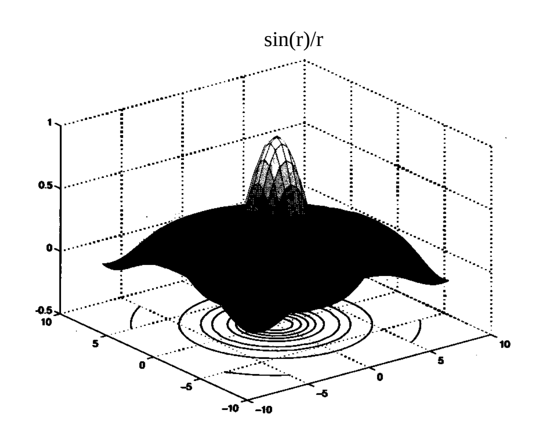


图13-33　下方有等高线的钟形表面图

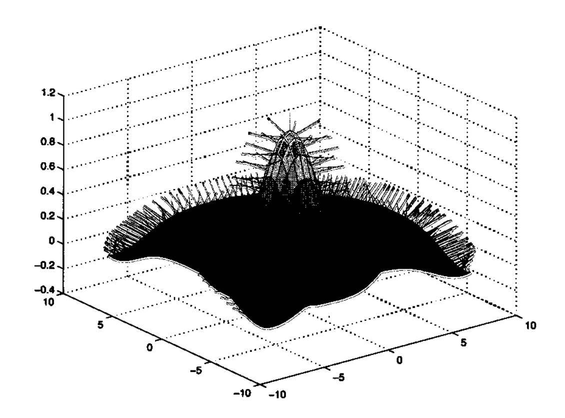
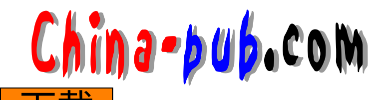


图13-34　函数的带有法线的钟形图

■



■ **例**13.21

下面研究一下对周围亮度、漫反射和单向反射使用不同权值的命令 surf1和surf。将使用常用于三维图形命令演示的内建函数 peaks。为了达到更好的视觉效果，按照第 14章所讲的对坐标轴进行处理，但是，在这里将省略一些专门的命令，而仅给出部分程序。给出如图13-35所示图形的程序如下：

```
[X,Y]      = meshgrid(-3:1/8:3);
Z          = peaks(X,Y).*sin(X);
[Nx,Ny,Nz] = surfnorm(Z);

s  = [-3 -3 2];              % 光源位置
k1 = [0,1,0,0];              % 漫反射
k2 = [0,0,1,1];              % 单向反射
DD = diffuse(Nx,Ny,Nz,s);

disp('Press a key after each plot.');

clf;

colormap(gray);
axis([-3 3 -3 3 min(min(Z)) max(max(Z))]);

surfl(X,Y,Z,s);         shading interp;  % 左上角
axis off; pause;

surfl(X,Y,Z,s,k1);      shading interp;  % 右上角
axis off; pause;

surfl(X,Y,Z,s,k2);      shading interp;  % 左下角
axis off; pause;

surf(X,Y,Z,DD);         shading interp;  % 右下角
axis off; pause;
```

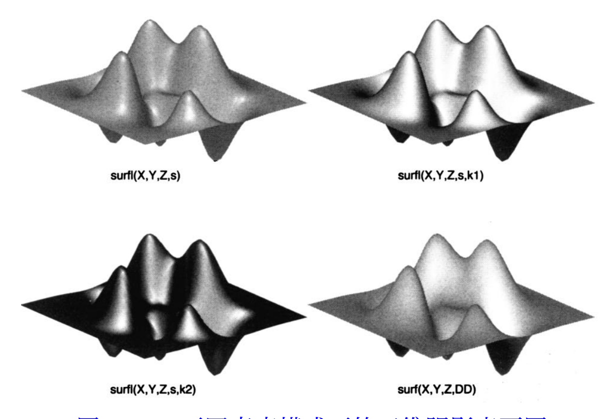
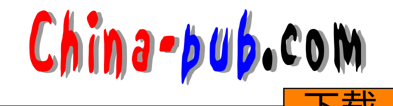


图13-35　不同亮度模式下的三维阴影表面图

■



从不同的角度观察，更容易观察一个图形。命令 view用于改变图的观察点，同时可以确定观察点、方位角和仰角；参见图13-36。还可以改变用命令viewmtx得到的视图。

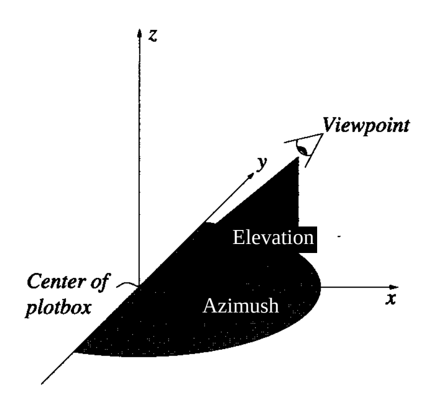


图13-36　命令view 示意图

命令集143　观察点和视图

`view(v,h)`	设置观察点角度。标量 *v*是方位角，即*xy*平面上逆时针得到的角度。平面上方的仰角由标量 *h*设定。*h*和*v*都用度来衡量。
`[v,h]=view`	返回命令view当前使用的*xy*平面内的角度*v*和*xy*平面上的角度*h*。
`view(r)`	设置观察点位置r=(*x y z*)。
`view(n)`	按下列值设置观察角度： *n*=2　标准二维观察点，从上直接向下。 *n*=3　标准三维观察点。
`view`	给出绘图时用于传递数据的4×4矩阵。
`view(T)`	MATLAB在绘图时使用4×4矩阵**T**。
`viewmtx(v,h,s,r)`	返回定义观察点和观察方向的4×4矩阵。使用help viewmtx　可获得更多信息。

■ **例**13.22

(a)假设图形窗口中有图13-33。命令`view([1 0  0])`将得到图13-37所示的同一个钟形表面图的侧视图。

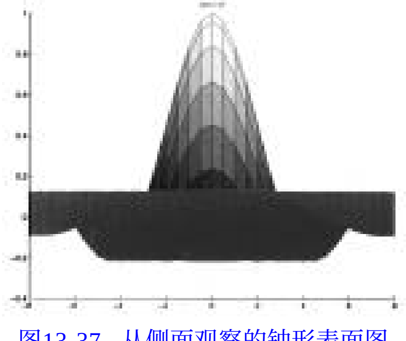
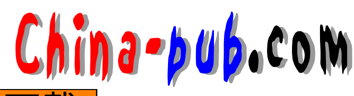


图13-37　从侧面观察的钟形表面图



(b) 矩阵**Matlabmatrix**由1和0组成，其中1形成字符MATLAB。由于矩阵太大，就不在这里列出。下列命令用来绘制出该矩阵的三维网格表面图和稀疏结构图。参见矩阵的 9.3节内容。这些命令保存在文件**meshplot.m**中：

```
% Matlabmatrix.m
% 创建一个由1和0组成的矩阵Matabmatrix, 代表MATLAB字符

Matlabmatrix;                              % 运行Matlabmatrix.m.

clf;                                       % 清除图形

subplot(2,2,1); mesh(Matlabmatrix);        % 绘制标准三维网格表面图
title('Standard view');                    % 写标题

subplot(2,2,2); mesh(Matlabmatrix);        % 绘制三维网格表面图
view([1 -4 2]);
axis([0 200 0 20 0 3]);                    % 调整观察点
title('Viewed from position [1,-4,2]');

subplot(2,2,3); mesh(Matlabmatrix);        % 等等
view([-1 -2 -7]);
title('Viewed from beneath');

subplot(2,2,4); spy(Matlabmatrix);
title('This is the matrix structure');     % 命令spy显示非零元素
```

用meshplot运行程序；得到图13-38。

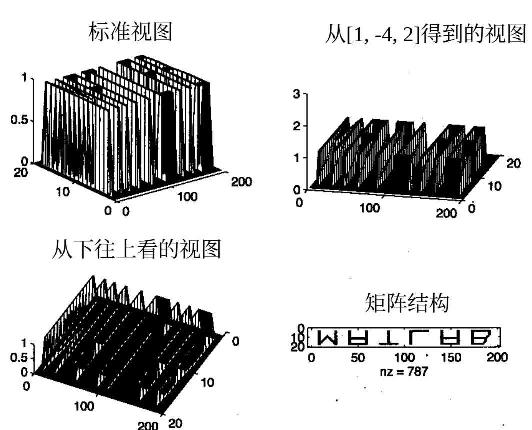


图13-38　用不同的方式给出的MATLAB矩阵图形

在使用命令spy(参见9.3节)时，要注意的是原点被放在左上角。这是描述矩阵最好的方法，因为，书写和定义矩阵就用的是这种方法。

(c) 尽管命令view最多的是用在与三维图形相关的操作中，但也可以将其用于二维图形。如果在图形窗口中已有了图 13-37，该图是用二维图形命令plot创建的，那么，命令view([1 0.6 0.35]) 将在三维空间中显示出观察点为(1, 0.6, 0.35)的同样的圆；如图13-39所示。

```
view([1 0.6 0.35])
```

命令surf和mesh可以用于在非矩形网格中绘制函数。在对图形程序的调用中必须包括坐标矩阵。

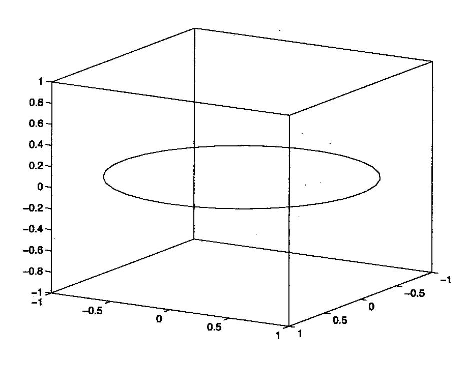


图13-39 空间圆

■ **例**13.23

假设要研究机翼周围部分的空气气压。这里的计算和网格的生成将远非文字所能描述，但是在MATLAB中很容易将其图形画出。网格存储在**X**和**Y**中，矩阵**P**表示空气气压。用axis命令改变图形大小，并使用view(2)命令从上方观察图形。使用命令fill对图形中的机翼进行染色。

```
% NACA0012式机翼周围的气流所形成的二维欧拉方程的解
% 攻击角度 alpna=0
% Mach 代码=0.85. Roe的秒序方法
% 网格= 200*80点，保存在FoilXY、mat中
% 计算结果存在Foil Pressure, mat的矩阵P中

if ~exist('X')
  disp('X does not exist.');
  load FoilPressure;                          % 载入求解的数据
  load FoilXY;                                % 载入画网格的数据

else
  disp('X exists');
end;

BodyX = X(:,1);
BodyY = Y(:,1);                               % 定义机翼
BodyU = P(:,1);

maxP = max(max(P)); minP = min(min(P));       % 最大和最小压力

% 机翼周围的网格
clf;
mesh(X,Y,P,ones(size(X))); view(2);           % 绘制网格
axis([-0.3 1.3 -0.3 1.3]);                    % 设定观察点
pause;

% 气压分布的表面图
clf;
surf(X,Y,P); shading interp; view(2);         % 绘制表面图
axis([-0.3 1.3 -0.3 1.3]);                    % 设定观察点
pause;

% 用黑色等高线和标注的等高图
clf;
fill(BodyX,BodyY,'k'); hold on;               % 绘制填充的机翼
axis([-0.3 1.3 -0.3 1.3]);                    % 设定观察点
[C,H] =  contour(X,Y,P,15,'k');               % 绘制等高线
```

```
h = clabel(C,H);                          % 等高线仰角标注

for lev = 1:length(h)                     % 改变标注字体
   set(h(lev),'fontname','times');
end;
pause;

% 用contourf和白色等高线填充等高图
% 每两条线做一个标注
clf;
DrawLevel = linspace(-1,1,30);            % 定义要画的线

[C,H] = contourf(X,Y,P,DrawLevel,'w');    % 绘制等高线
axis([-0.3 1.3 -0.3 1.3]);                % 设定观察点
pause;
```

结果如图13-40、图13-41、图13-23和图13-24所示。图13-40中的网格是用与P等大小的常数矩阵绘制的。

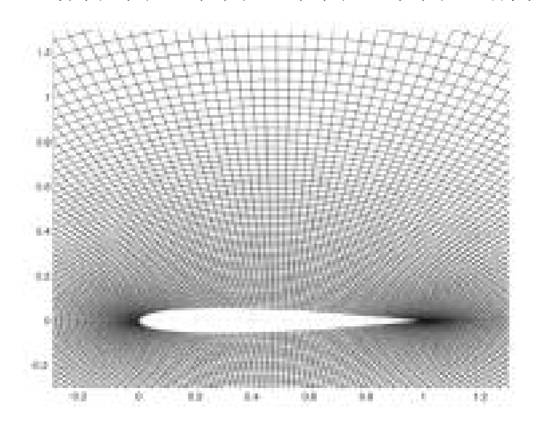


图13-40 机翼剖面周围的网格图形

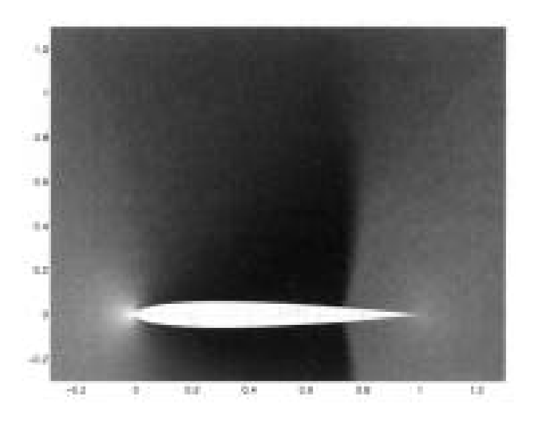


图13-41 机翼剖面周围的气压图形

■

camera是另一种更具体的控制观察点的方式。 camera的属性如图13-42所示。

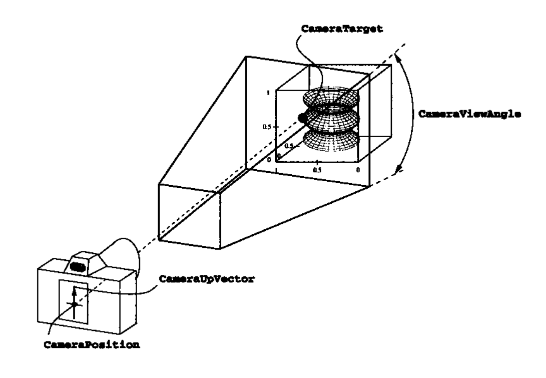
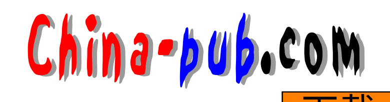


图13-42 观察点和它的属性

下列命令可用于控制观察点。



命令集144 观察点设置

命令	说明
camdolly(dx,dy,dz, *dirval, coordsys)*	按向量(*dx, dy, dz*)移动观察点。字符串**dirval**将决定观察目标是否随观察点一起移动。有两种选择： 'movetarget' 观察目标随观察点一起移动(缺省值)。 'fixtarget' 观察目标不随观察点移动。 字符串**coordsys**决定使用什么坐标系统。有三种选择： 'camera' 观察点中规格化的坐标系统，*x*轴指向右方，*y*轴指向上方（缺省值）。例如：camdolly(1,−1,0)向右下移动观察点，以便原观察目标移动到右上角的位置。 'pixels' 是基于像素的坐标系统，*x*轴指向右方，*y*轴指向上方。此时，变量*dz*将被忽略。 'data' 表示图形对象的坐标系统。
camlookat	移动观察点以观察绘制区域的某一特定部分。这需要了解有关各部分之间是如何联系的；参见第14章，特别是表14-12。
camorbit(dteta,dfi, *coordsys, dirval)*	绕观察目标旋转观察点。水平方向旋转 *delta*度，竖直方向旋转*dfi*度。字符串**coordsys**决定所使用的坐标系统，可以是'camera'(默认)或'data'(参见camdolly)。如果coordsys为'data'，则将绕从观察目标到*dirval*方向的连线旋转。
campan(dteta,dfi, *coordsys, dirval)*	绕观察点旋转观察目标。水平方向旋转 *delta*度，竖直方向旋转*dfi*度。字符串**coordsys**决定所使用的坐标系统，可以是'camera'(默认)或'data'(参见camdolly)。如果coordsys为'data'，则将绕从观察目标到*dirval*方向的连线旋转。
campos	返回观察点位置。
campos([x y z])	设置观察点位置。从现在起，观察点的位置将不再由MATLAB计算；见下面命令。
campos('pos')	决定是否由MATLAB自动计算观察点的位置。字符串 **pos** 可以是'auto'或'manual'。
campos('mode')	显示MATLAB是否计算观察点的位置。
camproj	返回当前投影。
camproj(*projec*)	设定当前投影。字符串 projec可以是'ortographic'(默认)或'perspective'。
camroll(dteta)	绕由观察目标到观察点的连线逆时针旋转观察点 *dteta*度。
camtarget	返回观察目标。
camtargert([x y z])	设定观察目标。从现在起，观察目标将不再自动计算；见下面命令。
camtarget('pos')	决定是否由 MATLAB计算观察目标。字符串 pos可以是'auto'或'manual'。
camtarget('mode')	显示MATLAB是否计算观察点的位置。

camup	返回向量up。
camup(up)	设定观察点的up向量，即：图中向上的方向，到向量up中。从现在起，up向量将不再自动计算；见下面命令
camup('pos')	决定是否由MATLAB自动计算up向量。字符串pos可以是'auto'或'manual'。
camup('mode')	显示MATLAB是否计算up向量。
camva	返回观察点的观察角度。
camva(val)	设置观察角度为val。从现在起，观察点的观察角度将不再自动计算；见下面命令
camva('pos')	决定是否由MATLAB自动计算观察点的观察角度。字符串**pos**可以是'auto'或'manual'。
camva('mode')	显示MATLAB是否计算观察点的观察角度。
camzoom(zoom)	改变图形大小。*zoom*取大于1的数，图形变大；取0～1之间的数，图形变小。
daspect	返回当前刻度。
daspect('pos')	决定是否由MATLAB自动计算刻度。字符串 pos可以是'auto'或'manual'。
daspect('mode')	显示MATLAB是否计算刻度。
daspect([x y z])	决定绘图区域的*x*，*y*和*z*方向上刻度标定方式。从现在起，刻度将不再自动计算；见下面命令。
pbaspect	返回当前刻度。
pbaspect('pos')	决定是否由MATLAB自动计算刻度。字符串 pos可以是'auto'或'manual'。
pbaspect('mode')	显示MATLAB是否计算刻度。

■ **例**13.24

再一次对图13-33进行观察。现在输入：

```
axis vis3d off          % 不绘制坐标轴
for x = 1:10
  camorbit(5,10);       % 绕观察目标旋转10次
  drawnow               % 画出图形
end
for x = 1:10
  camroll(5);           % 绕观察目标到观察点的连线旋转10次
  drawnow
end
```

首先，绕观察目标旋转观察点，接着绕由观察目标到观察点的连线逆时针旋转。最后的结果如图13-43所示。

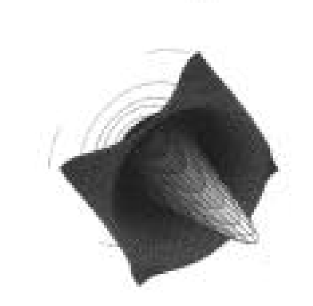


图13-43 旋转后的钟形表面图

■

MATLAB使用命令slice来以图形的方式研究三变量函数。该命令绘制三维表面图，并且图中各点的颜色与这些点的函数值是一致的。

命令集145 三维空间的部分图

slice(V,xs,ys, zs,nx)	绘制矩阵**V**定义的三变量函数的部分图。矩阵**V**本身是一个*nx*层的集合，对由带有三个参数的meshgrid得到的三个矩阵进行求值。向量**xs,ys**和**zs**规定要绘制的部分图形。
slice(x,y,z,V,xs ,ys,zs)	以矩阵**V**确定的颜色绘制三维平面。slice的旧语法仍然可用；这里是它的一个延伸。向量**x,y**和**z**用作坐标轴。参数**xs**、**ys**和**zs**确定绘图平面。

■ **例**13.25

下面来观察函数f$(x, y, z)=x^2+y^2+z^2$在立方体中的形状：

$[-1\ \ 1] \times [-1\ \ 1] \times [-1\ \ 1]$

首先要用meshgrid定义一个三维网格并对函数求值：

```
[X,Y,Z] = meshgrid(-1:0.1:1,-1:0.1:1,-1:0.1:1);
V = X.^2 + Y.^2 + Z.^2;
```

注意，在21^3个点上对函数进行求值。现在要决定绘制与坐标轴平行的那一部分图形。比如：向量(1 3 21)表示我们要绘制1、3和21部分。命令：

```
slice(V,[11],[11],[1 11]);
```

得到如图13-44的结果，它的部分图是由平面x=11, y=11, z=1和z=11定义的。

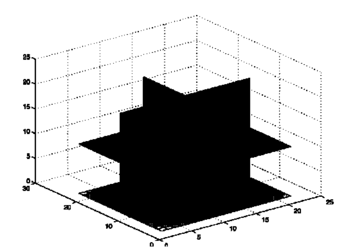


图13-44 用命令slice 得到的三变量函数图解

■

正如所预想的，函数值沿球形恒定。这在彩色图中比在黑白图中效果更加明显。

13.6　颜色控制

在MATLAB中，用户可以控制颜色和三维表面图的光照效果。

命令shading可以用来配置表面图的绘制方式。表面图可以在有网格或无网格情况下绘制，也可以在平面设置当前图形的阴影，或设置内插阴影。

命令集146　表面图属性

shading type	用下列属性重新绘制表面图：	
	faceted	设置表面阴影；这是缺省值。
	interp	设置内插阴影。
	flat	对平面设置当前图形的阴影。

在调色板一节的图P-3的Riemann表面图中给出了两种不同的阴影类型。

■ 例13.26

在例13.20中，绘制了一个网格可见的钟形表面图。用下面的命令可以在图形窗口中得到用内插值颜色着色的图13-33。

```
shading interp
```

得到如图13-45所示的结果。

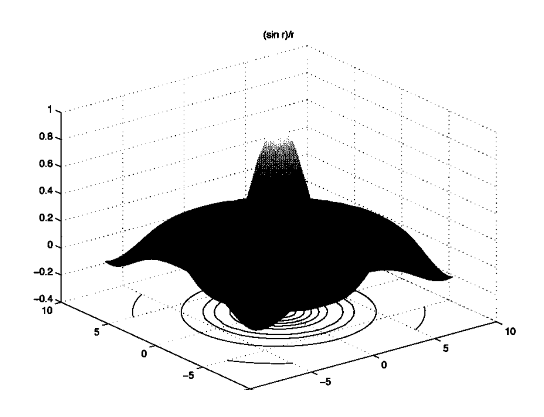


图13-45　用内插值颜色着色的钟形图

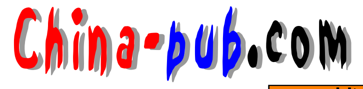
■

MATLAB使用色图来绘制表面图形。色图是一个$m \times 3$的矩阵，其中，行代表颜色，第1列给出红颜色的数量，第2列给出绿颜色的数量，第3列给出蓝颜色的数量，因此，该色图能给出m种颜色。

表面图的颜色由色图的下标来确定。下标通常与表面图的最大值和最小值有关。命令colormap常用于设置MATLAB使用的色图。

命令集147　色图

colordef definition	改变图形的颜色设置。**definition**的有效值为：



	white　采用亮度背景和黑色坐标轴。
	black　采用黑色背景和亮度坐标轴。
	none　采用MATLAB的颜色设置。
colormap(Cm)	设定当前颜色表为 **Cm**。矩阵**Cm**可以是MATLAB自身的色表，或用户定义的色表。
colormap	返回m×3矩阵的当前图形色表。
colorbar	在当前窗口中绘制竖直方向的条形颜色刻度。参阅colorbar。
colorbar('horiz')	在当前窗口中绘制水平方向的条形颜色刻度。

为了使用预先定义的色图，可以使用命令 colormap(winter(m))，其中m为色图中颜色的数目。MATLAB中共有17个预定义的色图(见表13-2和图P-1)。

表13-2　MATLAB**中的色图**

colorcube	返回RGB调色板中的均匀间距的颜色
lines	按照坐标轴的颜色顺序返回色图
autum	返回红色和黄色图
spring	返回红紫色和黄色色图
summer	返回绿色和黄色图
winter	返回蓝色和绿色图
gray	返回线形灰色刻度
hsv	返回从红到蓝再到红的深度颜色
hot	返回由黑到红到黄和白的混合的暖色
cool	返回青色和深红色的冷色
bone	返回带蓝色的灰色刻度
copper	返回铜色刻度
pink	返回粉红色的变化
flag	返回英制和美制标志的颜色，红色、白色、蓝色和黑色循环重复
prism	循环重复返回六种颜色：红色、橘黄色、黄色、绿色、蓝色和紫罗兰色
jet	返回一个交替的从红到蓝的色彩模式颜色表
white	返回一个全白的色图

■ **例**13.27

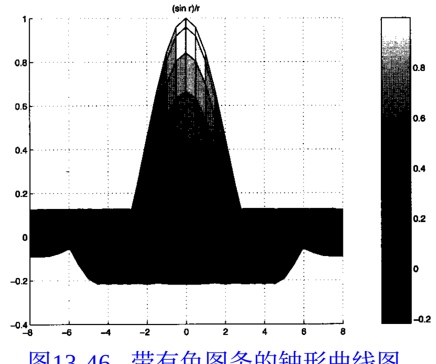


图13-46　带有色图条的钟形曲线图

假设图形窗口中已有了图13-37。命令：

```
colorbar
```

给出如图13-46所示的结果。

■

此外，还有几个对色图进行处理的命令。

命令集148 颜色处理

rgb2hsv(Cm)	返回$m\times3$的矩阵**Cm**中的rgb色图的色彩模型图。该色彩模型图包括与rgb图同样的颜色，但是颜色更深。
hsv2rgb(Cm)	返回色彩模型色图**Cm**的$m\times3$的rbg色图。
rgbplot(Cm)	绘制色图**Cm**各列的图形。线条分别由红色、绿色和蓝色绘制。
caxis(v)	设置色图的当前区间为**v**=$[v_{min}, v_{max}]$，其中v_{min}和v_{max}代表色图的较低和较高下标边界。
caxis	返回色图的当前区间。
caxis('auto')	恢复刻度为MATLAB自动标记的刻度。
spinmap(t,s)	用s步旋转色图t秒。如果s不确定，则令s=2，如果t不给出，
spinmap(inf)	则令$t\approx3$对色图做永久的旋转。
brighten(s)	如果$0<s<1$，则加亮当前色图。如果$-1<s<0$，则加暗色图，重画图形。
nt=brighten(Cm,s)	返回变亮或变暗的**Cm**色图，但是不重画当前图形。
contrast(Cm,m)	返回从颜色表**Cm**中长度为m，并且增加黑白监视器对比度的颜色表。如果省略m，将返回与**Cm**同样大小的色表。
whitebg	在黑白之间切换图形窗口的背景颜色。如果有必要，刻度颜色等将改变为可见的。
whitebg(str)	根据**str**设置背景颜色，可以是字符串(参见表13-1)或rgb向量。
graymon	对黑白监视器设置参数。

■ **例13.28**

用命令rgbplot对色度模型色图做一下研究：

```
clf;
rgbplot(hsv);
title('rgbplot of hsv');
axis([0 70 -0.1 1.1]);
```

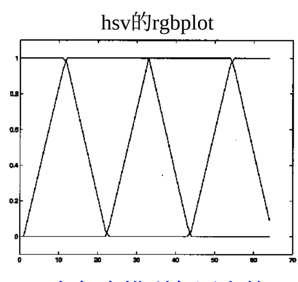
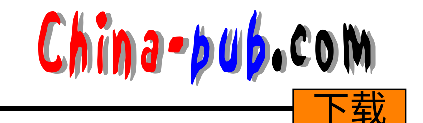


图13-47 一个色度模型色图上的rgbplot图形



得到图13-47的结果。

■

13.7 图形窗口的硬拷贝

假设已安装了必要的硬件和软件，就可以从 MATLAB的图形窗口中得到图形的一个硬拷贝。命令print可用于硬拷贝到文件中或送打印机，应用于当前图形。在 PC或Macintosh系统中，文件菜单下的打印选项是打印图形的最简单方式。

命令集149 **打印硬拷贝图形**

print	将当前图形窗口的一个高分辨率拷贝发送至打印机。这要求将print命令分配给打印机，输入help print 可获得更多信息。
print filename	将当前图形窗口的拷贝保存到文件 **filename**中。
print filename −deps	将副本以eps格式，即压缩的附录，保存到文件**filename**中。键入help print会有更多的选项。
[str,dev]=printopt	给定print使用的命令字符串和设备。这有可能要修改这个M文件；输入help printopt可获得更多信息。

如果打印机在安装时并未指定给MATLAB，那么，可以先将图形存成一个文件，然后使用系统命令将该文件发送至打印机。比如：在SUN公司的Solaris2.x网络操作系统上的一种类型为：

```
!lp -dskrivarnamn filename.ps
```

命令orient可以用来设定硬拷贝图形的打印方向。如果要改变方向，该命令设定的方向将优先于用print命令的设定方向。

命令集150 **纸张方向控制**

orient landscape	设定下一次打印的方向为landScape，即：水平方向。
orient portrait	设定下一次打印的方向为portrait, 即：竖直方向。
orient tall	设定下一次打印的方向为竖直方向，并将刻度设为全部纸面。
orient	将当前方向返回到一个字符串中。

■ **例**13.29

本书包含许多来自MATLAB的图形。这些图形都是由各种 MATLAB命令创建的，并且可以用类似下面的语句制作硬拷贝：

```
print -deps fig10
```

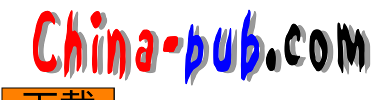
这些图形直接引入到本书中。这是MATLAB结合其他程序编程的例子。

■

orient命令可以设置当前图形的属性。图形对象的使用 (参见第14章)给出了硬拷贝图形属性的更具体的控制。纸张的大小、纸张中的位置、背景颜色和一些其他属性都可以设定。

13.8 声音

MATLAB可以使用sound命令制作声音向量。



命令集151 声音

`sound(y)`	将向量**y**传送给扬声器。向量确定了最大振幅。
`sound(y,f)`	与上面的命令相似，而且还设定采样频率为 *f* Hz。该命令不能用于SUN公司的SPARC工作站。
`soundsc(x, f,slim)`	采用与`sound`相同的方式播放向量x，除了`soundsc`给出的声音向量，声音可以尽可能地大。如果给出 *f*，*f*就表示采样频率。这里**slim**设定满音量范围，缺省值为[*min*(*x*) *max*(*x*)]。

■ **例**13.30

(a) 正弦波听起来是这样的：

```
x=sin(linspace(0, 10000, 10000));% 一个纯正弦波。
sound(x);                        % 产生声音。
```

(b) 可以用`load`命令载入几个预定义的声音。这里，就试试其中的两个。

```
load train;                       % 装入产生火车汽笛的声音数据。
whos;                             % 显示变量y, Fs：向量y:表示创建的信号；标量Fs:
                                  表示以Hz为单位%的频率。
sound(y);                         % 产生声音。
load chirp;                       % 装入产生鸟儿唧唧喳喳声音的数据。
sound(y);                         % 产生新的声音。
```

在某些系统中还有一些附加的声音命令；用`help sounds`可以了解这些特殊的系统。SUN公司的SPARC工作站使用存成mu-law编码数据的声音向量作为声音文件。

命令集152 SPARC工作站上的声音命令

`auread(fstr)`	从文件**fstr**中读取并返回一个向量。
`auwrite(sv,fstr)`	将向量**sv**写入Sun的音频文件**fstr**中。
`lin2mu(sv)`	将线性声音向量**sv**转化为mu-law编码的向量。
`mu2lin(sv)`	将mu-law编码的向量**sv**转化为线性声音向量。

微软Windows操作系统中使用的声音文件为.wav格式。

命令集153 专用于微软Windows操作系统的声音命令

`wavread(fstr)`	在文件**fstr**中返回采样数据。用`help wavread`可获得更多信息。
`wavwrite(sv,f,fstr)`	以采样频率*f*将采样声音向量**sv**写入文件**fstr**中。

第14章　高 级 图 形

MATLAB的图形系统是面向对象的，也就是说图形的输出，如曲线，是建立图形对象。通常用户不必去关心这些高级 MATLAB命令包含的对象。然而有时为了调整对象也要用一些低级的MATLAB命令。

MATLAB中介绍了图形用户界面 (GUI)的应用，如单选按钮、滑标和菜单。利用这些用户能够很容易地进行图形控制。

在MATLAB中加入一系列的图片就可以创建出动画来，利用这些动画可以做一些有趣的演示。

14.1　图的结构

一个图形是由许多的**图形对象**组成的，这些对象是以层次顺序保存的。举一个例子来说明它们之间的关系。

■ **例**14.1

用不同线型绘制图形的例程。

```
clear;
x=0.1:0.1:4*pi;              % 生成向量x。
y1=sin(x);                   % 生成y1值。
y2=sin(x) ./x;               % 生成y2值。
figure;                      % 创建一个新窗口。
subplot(1, 2, 1);            % 定义第一个子图区域。
plot(x, y1);                 % 用实线画曲线。
subplot(1, 2, 2);            % 定义第二个子图区域。
plot(x, y2,'*');             % 用'*'号画曲线。
```

上述命令在图形窗口的两个子域内画出了关于 x的两个函数y1和y2的图形。结果如图 14-1所示。

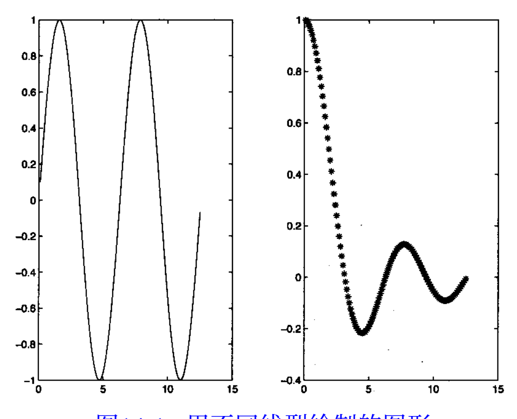


图14-1　用不同线型绘制的图形

■

一个图形由安排在一个层次结构上的五个图形对象组成。首先有一个窗口，这是一个**图形对象**。然后由两个**轴对象**来定义相应的坐标轴。这些都可以用 subplot命令来完成。最后，用plot命令来创建两个**线条对象**。例14.1中的对象层次结构在图14-2中给出。

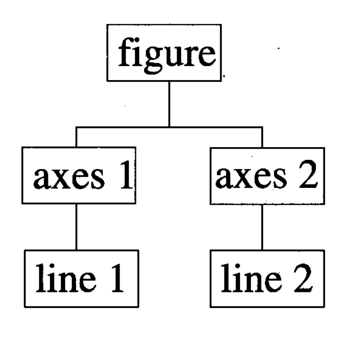


图14-2　对象层次结构

对于每一个对象都可以修改它的一些属性。例如，可以改变图形窗口的位置和图形对象的背景色。对于一个轴对象可以改变它在图形区域内的刻度大小和位置。线条对象可以变得更细，改变成另一种颜色，另一种线型等等。

由于是层次结构，所以某个对象改变时，会影响到这个结构中它以下的所有对象。如果使用鼠标改变图形对象的屏幕位置，线条和轴对象也会跟着变。但是如果改变右边轴对象的轴刻度，那么只影响这个轴上的线条。

图14-2中还少一个对象：**根对象**。它是所有图形对象的根，也就是整个层次结构的根。

在创建对象的同时可以直接修改一些对象的属性。在画 **y2**图形时，‘*号表示用星号来画曲线而不是用实线。调用 subplot可以规定图形区域在窗口的位置。然而还有许多属性只能用在后面描述的set命令来改变。下面的例子修改了上例中图形的部分属性。

■ **例**14.2

```
% 在上例中，改变左边子图区域的位置和第2个子图的X轴坐标

clear;

x  = 0.1:0.1:4*pi;
y1 = sin(x);
y2 = sin(x)./x;

fg = figure;                              % 创建窗口和图句柄

r1 = subplot(1,2,1);                      % 创建子图和轴句柄
l1 = plot(x,y1);                          % 创建线条和线条句柄

r2 = subplot(1,2,2);
l2 = plot(x,y2,'*');

disp('The previous example');
pause;

set(r1,'Position',[0.1 0.1 0.3 0.3]);    % 改变位置
set(l1,'LineWidth',5);                    % 加粗线条

set(r2,'XTick',[1 4 11]);                 % 改变X轴
set(l2,'LineStyle','+');                  % 改变线型

pause;
delete(fg);                               % 删除窗口
```

在使用命令figure、subplot和plot时定义一个变量名来创建对象的**句柄**或者**标识**。通过使用句柄可以用set命令来修改对象属性。在本例中修改对象的一个属性，改变了第1个子图的位置和第2个子图中的*x*轴刻度值。现在得到的窗口如图14-3所示。

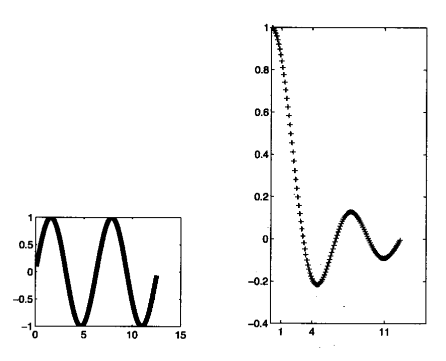


图14-3 对图14-1修改一些属性后的图形

■

14.2 图形对象

MATLAB 5.2中的各种对象都列在表14-1中。

在图14-4中给出了对象的层次结构图。

在图14-5中给出了不同图形对象的例子。

父对象影响它所有的**子对象**，这些子对象又影响它们的子对象，以此类推。结果是轴对象会影响像对象，但不影响用户界面控制。

根据表14-1可知，根对象的句柄值是零，而图形对象的句柄值是整数，其他对象则用浮点值作为句柄值。

画一个对象，可以使用和对象名字的相同低级命令。如画一线条，可以用命令 line。

对象的属性有两类：

• 属性，用来决定对象的显示和保存的数据。

• 方法，用来决定在对对象操作时调用什么样的函数，如当创建或者删除对象时，或当用户点击它们时。

一些属性有缺省值，如果没有特殊说明，就是用这些缺省值。

有一些属性是用来规定对象色彩的，它们以RGB三元组的形式给出，也就是说，用一个有三个元素的向量[*r g b*]($0 \le r, g, b \le 1$)来表示颜色中的红、绿和蓝色，例如，用[1, 0, 0]表示红色。也可以用预定义在MATLAB中表示颜色的字符串来代替RGB三元组，如‘black’和‘blue’。

在helpdesk中的句柄图形对象给出各种不同类型对象的详细说明。 MATLAB手册《使用MATLAB图形和用MATLAB建立GUI》也可得到相关的信息。

表14-1 图形对象

对 象	父 代	描 述
root	—	屏幕是一个根对象。所有其他的图形对象都是根的子对象。根对象的句柄值是零

（续）

对　　象	父　　代	描　　述
figure	root	屏幕上的窗口是一个图形对象，对象的句柄值是整数，在窗口的标题中给出
axes	figure	轴对象在窗口中定义一个图形区域。可以用来描述子对象的位置和方向
uicontrol	figure	用户界面控制。当用户用鼠标在控制对象上点击时，MATLAB会完成一个相应规定的任务
uimenu	figure	创建一个窗口菜单，用户用这些菜单能够控制程序
uicontext-menu	figure	创建一个图形对象的快捷菜单。也就是当用户点击图形对象时会显示出菜单来
image	axes	用当前的色图矩阵定义一个图像。图像可以有自己的色图
line	axes	用plot、plot3、contour和contour3创建一些简单的图形
patch	axes	创建补片对象
surface	axes	输入定义一个有四个角的曲面，可以用实线或内插颜色来绘制，或者作为一个网格
text	axes	字符串，它的位置由它的父对象——轴对象来指定
light	axes	定义多边形或者曲面的光照

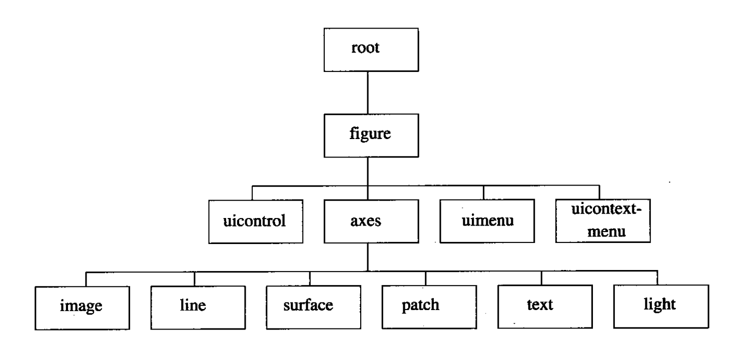


图14-4　图形对象层次结构图

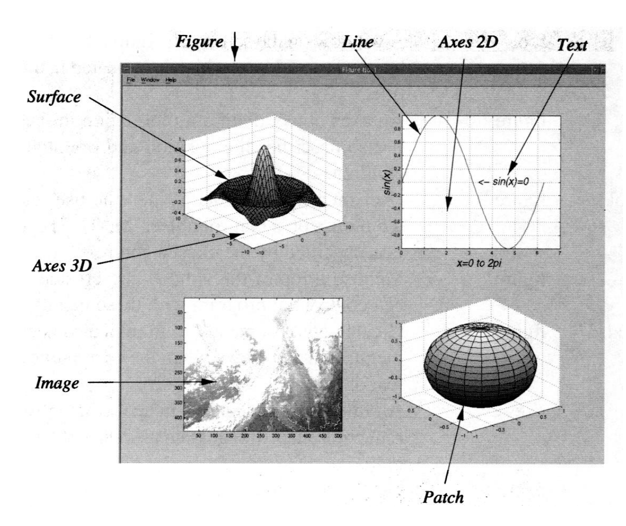
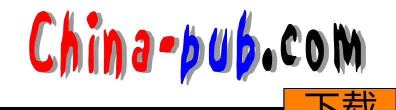


图14-5　不同类型图形对象



14.2.1 通用函数

MATLAB有两个基本命令get和set用来处理图形对象。通过使用它们，可以给出和修改所有对象的属性值。

■ **例**14.3

(a) 创建一个图形窗口，它的句柄是*gfp*：

```
gfp=figure;
```

先查看窗口使用的单位类型：

```
get(gfp,'units')

ans =
      pixels
```

现在来改变窗口的大小和位置：

```
set(gfp,'Position',[100 500 400 400])
```

窗口的左下角移动到点(100，500)处，大小设置为400×400像素点。

(b) 有些高级命令，如plot可以直接改变这些属性值。例如，例14.1中的曲面可以这样来画：

```
plot(x,y1,'LineWidth',5);
plot(x,y2,'LineStyle','+'); or plot(x,y2,'+');
```

(c) 如果想知道在set和gcf之间有哪些指针可以挑选使用；见命令集155。

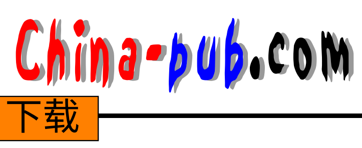
```
set(gcf,'Pointer')

[ crosshair | fullcrosshair | {arrow} | ibeam | watch | topl |
topr | botl | botr | left | top | right | bottom | circle |
cross | fleur | custom ]
```

■

命令集154　通用对象函数

set(h,prstr,alt,...)	对向量**h**指向的对象设置其属性**prstr**为值*alt*，在它们之后还可以有几个属性。
set(h,a)	对向量**h**指向的对象设置其属性值。参数**a**是一个结构，结构中的域名是对象中可改变的属性名，域的值是赋予相应属性的新值。
set(h,pn,pv,...)	对向量**h**指向的对象设置其属性值。参数**pn**是一个包含可改变的属性名的细胞向量，**pv**是一个包含设置新值的细胞向量。在它们之后还可以有几个属性的细胞向量。
set(h,pn,pv)	对**h**(*i*)指向的对象设置名为**pn**(*j*)属性的值为**pv**(*i*, *j*)。参数**h**是一个包含对象句柄的向量，**pn**是一个包含属性名的细胞向量，**pv**是一个包含有*m*×*n*个值的细胞矩阵，*m*是**h**的长度，*n*是**pn**的长度。这个变量可以单独地设置对象属性。
set(h)	将句柄*h*的对象所有可改变的属性返回在一个结构中，结构的域名是这些属性的名字，域值是相应的属性值。
set(0,'Default')	检查对象属性的缺省值。返回一个结构，域名为



	'ObjectNamePropertyName', 域值是相应的属性缺省值。注意：只返回值为字符串的属性。
set(0,'DefaultObjectNamePropertyName')	返回一个细胞向量，它包含有对象名为ObjectName、属性名为PropertyName的缺省值。注意：只返回值为字符串的属性。
set(h,prstr)	返回一个细胞向量，包含句柄 *h*的对象中属性名为 prstr的所有可能值。
get(h)	列出句柄*h*的对象所有属性和相应的值。所有这些都返回在一个结构中，域名为属性的名字，域值为相应的属性值。
get(h,prstr)	返回句柄*h*的对象且属性名为prstr的当前值。
get(h,pn)	返回一个细胞矩阵，元素 (*i*, *j*)的值是句柄h(*i*)的对象且属性名为pn(*j*)的属性值。参数pn是一个细胞向量，h是一个包含句柄的向量。
get(0,'Factory')	返回所有对象类型的所有属性的厂家设置值，用户可以改变它们的缺省值。厂家设置值返回在一个结构中，域名为'ObjectNamePropertyName'，域值为相应属性的厂家设置值。
get(0,'FactoryObjectNamePropertyName')	返回对象ObjectName中属性PropertyName的厂家设置值。
get(0,'Default')	检查有缺省值的对象。结果返回在一个结构中，域名为'ObjectNamePropertyName'，域值为相应属性的缺省值。
get(0,'DefaultObjectNamePropertyName')	返回对象ObjectName的属性PropertyName的缺省值。
allchild(h)	返回句柄*h*(*i*)的对象中所有子对象的句柄。隐含句柄也返回，见表14-2中'HandleVisibility'。如果h是一个标量句柄，则返回得到一个向量。如果h是一个向量，则返回一个细胞矩阵。
findobj	返回一个根对象和它的所有子对象的句柄向量。
findobj(h,prstr,*alt*,...)	如果给出参数**prstr**和*alt*，则返回一个向量，包含将属性**prstr**设置为*alt*的所有对象的句柄。在它们之后还可以写上几个属性。如果给出参数 **h**，就限制在**h**中的对象和它们的子对象中搜索。如果没有给出 **prstr**和*alt*，则返回一个包含向量h中的对象的子对象句柄向量。
findobj(*h*,'*flat*',prstr,alt,...)	同上，但是不检查对象的子对象。
findall(h)	和findobj(h)相似，而且还返回子对象的隐含句柄，见表14-2中'HandleVisibility'。
copyobj(h,p)	复制对象和它们的子对象。向量**h**中包含要复制的对象句柄，向量**p**中包含的它们父对象的句柄。这些父对象的类型必须和原来的父对象相同。函数返回一个包含新对象的句柄向量。
ishandle(h)	检查向量**h**中的元素是否是对象的句柄。函数返回元素是 0



	和1的向量。如果h中的元素是一个对象的句柄，则在向量的相应位置为1；否则为0
`setuprop(h,prstr,val)`	用户可以对图和轴类型的对象定义新的属性。在字符串**prstr**中给出句柄为*h*的对象新属性，属性值设置为值 **val**，**prstr**和val可以为任何值。除非属性 **prstr**已经存在，否则就新建。
`getuprop(h,prstr)`	获取句柄为*h*的图形对象或者轴对象中用户自定义的属性**prstr**的属性值。
`clruprop(h,prstr)`	删除句柄为*h*的图形对象或者轴对象中用户自定义的属性**prstr**。
`handle2struct(h)`	将向量**h**中的句柄对象层次变换成一个结构，这个结构有如下的域： **.type** 对象的类型。 **.properties** 包含属性值的一个结构。 **.children** 一个结构矩阵，元素是句柄向量**h**中的对象的每个子对象。 **.handle** 对象的句柄。 **.special** 非文件式成员。
`struct2handle(s)`	命令handle2struct的逆操作：将结构s变换成对象的层次。

MATLAB 5有一个名为propedit的工具，是属性编辑器。这个编辑器比低级命令 set和get用起来更方便。在建立图形用户界面时，这个工具显得更加有用，见命令集 170。

而且，还有三个函数可以获取当前图形对象的句柄。

命令集155　当前对象

`gcf`	获取当前图形对象的句柄。
`gca`	获取当前轴对象的句柄。
`gco(h)`	获取当前对象的句柄，也就是鼠标最后点击的对象。如果给出了*h*，则获取句柄为*h*的窗口中当前对象的句柄。

■ 例14.4

下面的命令用来获取当前窗口和轴的属性：

```
get(gcf);          % 获取当前图形对象的属性。
get(gca);          % 获取当前轴对象的属性。
```

■

■ 例14.5

也可用subplot命令移动对象。假设有两个子图，一个是牙刷，另一个是牙膏，如图14-6所示。要获取下图的位置，可以输入：

```
subplot(2,1,2);
get(gca,'Position')

ans =
        0.1300   0.1100   0.7750   0.3400
```

移动对象，可以输入：

```
set(gca,'Position',[0.15 0.49 0.775 0.34])
```

得到图14-7。这样轴对象就放在另一个图的上面。



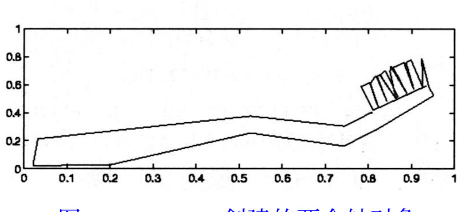


图14-6 subplot创建的两个轴对象

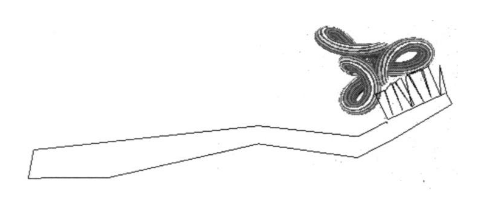


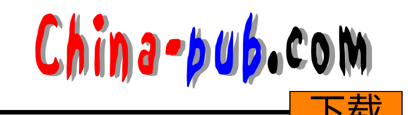
图14-7 将牙膏放在牙刷上的操作

■

还有一些函数用来删除对象、恢复缺省值和保存对象到文件中。

命令集156　其他通用函数

clf	清除当前窗口。
clf reset	清除当前窗口，除Position、Units、PaperPosition和PaperUnits外，重新设置所有图形对象的属性，见表14-6。
cla	删除当前坐标系。
cla reset	删除当前坐标系。除Position和Units外，重新设置所有轴对象的属性，见表14-10。
rotate(h,ax,a,*o*)	计算对象*h*中角*a*绕轴**ax**旋转的角度，可选参数 **o**给出旋转的起始位置，缺省值为[0 0 0]。
remapfig(pos, newpos,fig,*h*)	将子对象移动到具有句柄*f i g*的图形对象中。由**pos**规定内部区域大小的每个子对象移动到 **newpos**规定的新位置。参数**pos**和**newpos**都是这样的形式[*left bottom width height*]。一个轴对象的整个区域为 [0 0 1 1]。如果没有指定*fig*，就使用缺省值。如果给出**h**，则只移动向量**h**中的子对象。
reset(h)	恢复具有句柄*h*的窗口(图形对象)或者坐标系(轴对象)的缺省值。
delete(h)	删除具有句柄*h*的对象。
close(h)	关闭当前窗口。如果给出 **h**，则关闭**h**中的窗口。参数**h**可以是一个标量，也可以是包含有轴对象句柄的向量或者矩阵。
close name	关闭名为**name**的窗口。



`close all`	关闭所有没有隐含句柄的窗口，隐含句柄可通过属性 HandleVisibility来设置，见表14-2。
`close all hidden`	关闭所有窗口，包括有隐含句柄的窗口。
`status=close(···)`	如果相应的窗口已关闭，则令**status**为1；否则为0。
`hgsave(h,filename)`	将具有句柄*h*的对象及其他的子对象保存到文件 **filename**中。如果**filename**缺少后缀名，则加上.fig。
`hgload(filename)`	从文件**filename**中读取对象和它的子对象(如果有的话)。如果**filename**缺少后缀名，则加上.fig。最后返回读取对象的句柄。

14.2.2 共有属性和方法

由于继承，对于所有类型的对象有一些相同的属性和方法。然而这些当中有一部分对有些对象来说是没有意义的；见属性表和每种类型对象的方法。

表14-2 共有属性和方法

ButtonDownFcn	当对象被鼠标选择时，返回MATLAB回调字符串
Children	对象的所有子对象句柄的向量
Clipping	数据限幅模式，‘on’(缺省值)：只显示在坐标轴界限内的部分图形对象；‘off’：没有这个限制，也显示坐标轴外的部分
CreateFcn	决定用什么样的M文件或者MATLAB命令来创建对象。这必须用缺省值，例如创建一个图形对象： set(0,'DefaultFigureCreateFcn','function') 其中字符串function是M文件名或者MATLAB命令
DeleteFcn	决定删除对象时运行的M文件或者MATLAB命令
BusyAction	MATLAB处理对象的回调函数中断方式。如果将 Interruptible(见下面命令)，设置为‘off’，BusyAction可以有下面几种情况： ‘queue’ (缺省值)将回调函数的中断请求放入一个挂起队列中直到对象的回调函数完成 ‘cancel’ 忽略其他回调函数所有可能的中断
HandleVisibility	对象的子对象列表中的对象句柄是否可访问；见Children ‘on’ (缺省值)总是可访问 ‘callback’ 只有回调函数或者调用回调函数的函数可以访问，这样防止用户从命令行中对对象进行修改 ‘off’ 不可访问
HitTest	对象是否被鼠标选中，也就是这个对象是否为当前对象。HitTest可以设置为‘on’(缺省值)或者‘off’。也可参见命令集155中的命令gco和表14-7中的轴对象属性CurrentObject。
Interruptible	指定对象回调字符串是否可中断。如果Interruptible是‘on’(缺省值)，则该对象回调函数可以被其他回调中断。如果Interruptible是‘off’，则该对象回调函数不能被其他回调中断
Parent	对象的父对象句柄

(续)

Selected	对象是否被选中，值可以为‘on’或者‘off’
SelectionHighlight	在屏幕上选中的对象是否有四个边句柄和四个点句柄。值可以为 ‘on’(缺省值)或者‘off’
Tag	用户用来标识对象的字符串，在建立图形接口时这很有用
Type	只读对象类型的字符串
UserData	是一个矩阵，包含有用户要在对象中保存的数据。矩阵不被对象本身使用
UIContextMenu	和对象相联的快捷菜单句柄。当在对象上按下鼠标右键时， MATLAB显示出快捷菜单。见14.2.8节
Visible	控制对象在屏幕是否可见。值可以为‘on’(缺省值)或者’off’

14.2.3 根对象

上面提到屏幕是一个根对象，它是所有其他对象的父对象。 MATLAB中可以用set和get设置和获取它的一些属性。根对象有如下的属性和方法。

表14-3 根对象的常用属性和方法

Automatic-FileUpdates	值可为‘on ’或者‘off ’，它的使用是非文件式的
CallbackObject	获取正在执行的回调函数的对象句柄。如果没有回调函数在执行，则返回[]。也可参见命令集171中的命令gcob
Language	系统环境变量，它的使用是非文件式的
CurrentFigure	获取当前图形对象的句柄。如果不存在图形对象，则返回 []。也可参见命令集155中的命令gcf
Diary	一个决定是否将命令窗口中的输入输出存入文件的字符串。值可为 ‘on ’或者‘off ’(缺省值)
DiaryFile	一个表示保存输入输出文件名的字符串，见 Diary
Echo	在文件执行时，是否显示M文件的每一行。值可为‘on‘或者‘off ’(缺省值)
ErrorMessage	包含MATLAB最近一次错误信息的字符串
Format	控制MATLAB数字显示格式的字符串。可选择的值有：‘short’、‘shortE ’(缺省值)、‘shortG’、‘long’、‘longE’、‘longG’、‘bank’、‘hex’、‘+’和‘rational’
FormatSpacing	控制MATLAB在命令窗口中输出间隔行的字符串。值可以为 ‘loose ’(缺省值)和‘compact’
PointerLocation	相对屏幕左下角指针的当前位置向量。
PointerWindow	含有鼠标指针的图形只读句柄。如果鼠标不在任何 MATLAB图形窗口中，则值为0
Profile	决定是否使用profile工具的字符串。当程序执行时，这个工具可以测出程序的不同部分运行所需的时间。值可以为 ‘on’或者‘off’(缺省值)
ProfileFile	控制profile数据保存到什么样的文件中(见上)
ProfileCount	是一个包含profile数据的$n \times 1$向量。向量中的元素k是M文件在运行时在k行找到profile工具的次数
ProfileInterVal	在一定时间间隔内， profile工具检查正在运行的脚本文件的行数
RecursionLimit	限定嵌套递归的次数。如果超过这个次数， MATLAB会给出一个错误信息
ScreenDepth	整数，指定以比特为单位的屏幕颜色深度



(续)

ScreenSize	有四个元素的只读位置向量[*left bottom width height*]，指定屏幕大小尺寸
ShowHidden-Handles	控制所有对象的所有句柄的可访问性，使各自的HandleVisibilty属性失效；见表14-2 属性值可以为‘o n或者‘off’(缺省值)
Units	大小和位置的度量单位，可选下列单位： ‘pixels’(标准)　屏幕像素 ‘normalized’　归一化坐标 ‘inches’　英寸 ‘centimeters’　厘米 ‘points’　排字机的点，等于0.353毫米 ‘characters’　字符
Parent	总是空矩阵[]，　根对象没有父对象

MATLAB在X Windows环境下使用时，根对象还有一些另外的属性和方法。

表14-4　在X Windows环境下根对象的另外属性和方法

TerminalHideGraphCommand	文本串，从图形转换到命令模式时，隐含图形的命令序列
TerminalShowGraphCommand	文本串，从命令转换到图形模式时，显示图形的命令序列
TerminalOneWindow	终端有一个窗口，值为‘on’(缺省值)；终端有多个窗口，值为‘off’
TerminalDimensions	终端尺寸大小，以像素来衡量
TerminalProtocol	终端类型模式设置，可选的值为： ‘none’　非终端模式，不连到X服务器 ‘x’　找到X服务器，X Windows模式 ‘tek401x’　Tektronix 4010/4014仿真模式 ‘tek410x’　Tektronix 4100/4105仿真模式

下列的常用属性和方法对根对象是没有意义的。

表14-5　根对象是没有意义的属性和方法

BusyAction	ButtonDownFcn	Clipping	CreateFcn
DeleteFcn	HandleVisibility	HitTest	Interruptible
Selected	SelectionHighLight	UIContextMenu	Visible

■ **例**14.6

根对象的句柄是0，可以象这样来调用：

```
scrsize = get(0,'ScreenSize')

Scrsize =
          0     0     1152     900
```

命令返回得到屏幕大小的像素值。

■

14.2.4　图形对象

图形对象就是一个图形窗口，它的父对象是屏幕，即根对象。因此图形对象继承了根对

象的许多属性。属性可以在图形对象创建时修改，也可以用 set命令来修改。

创建图形对象有下列几种方式。

命令集157 图形对象

figure(prstr,*alt*,...)	设置属性**prstr**的值为*alt*，并返回一个图形对象的句柄。还可以再带几个属性及其相应的值。
figure(f p)	设置句柄为*f p*的窗口为当前图形对象。所有的图形命令都可用在当前图形中。
refresh(*f p*)	重新画句柄为*f p*的图形。如果只是给出refresh，则重画当前图形。
drawnow	强制MATLAB画一个对象。例如，在循环语句中用 plot命令画对象时。如果不在每个plot命令后都给出drawnow命令，就只有在循环结束才画出对象。
newplot	打开一个新图形或者坐标系；是否清除当前图形或者轴取决于属性NextPlot的属性值。用help newplot可以得到更多信息。
fignamer(str)	得到以字符串**str**开始的下一个空闲图形的名字。这样就可以用来设置下面的Name属性。
setptr(h,cursor)	设置句柄*h*的图形指针形状。指针形状由**cursor**给出，用help setptr可以知道有哪些可选的形状。返回一个指针形状的细胞矩阵，也可参见下面的各种Pointer属性。
p=getptr(h)	返回得到一个包含句柄*h*的图形对象指针形状的细胞矩阵**p**。这个细胞矩阵可以用来设置指针形状：setptr(h,p{:})。也可参见下面的各种Pointer属性。

图形对象有下列一些常用属性和方法。

表14-6 图形对象的常用属性和方法

BackingStore	当图形窗口移动到屏幕前时，为了快速刷新屏幕，决定是否存储它的拷贝。它的值可以为‘on’(缺省值)或者‘off’
IntegerHandle	控制句柄是可用的整数值还是不可用的浮点数值。如果是整数，IntegerHandle设为‘on’(缺省值)，否则设为‘off’
MenuBar	控制MATLAB菜单在图形窗口的顶部显示。值可以为‘figure’(显示菜单，缺省值)或者‘none’(隐藏菜单)
Name	图形窗口的标题，缺省值为空字符串
NextPlot	决定在图形窗口中新图的绘制方式： ‘add’ (缺省值)在当前图形窗口中加上新的对象 ‘replace’ 在画图前，将除位置属性外的所有图形对象属性重新设置为缺省值，并删除所有子对象。等同于命令clf reset ‘replace children’ 删除所有子对象，但是不重新设置属性值。等同于命令clf
NumberTitle	在图形标题中加上图形编号 如果NumberTitle设为‘on’(缺省值)，窗口标题是“FigureNo.N: Name”，其中Name(见上)是一个字符串。如果NumberTitle设为‘off’，窗口标



(续)

	题仅仅为“Name”
Pointer	鼠标指针形状，可选的形状有下列几种：‘crosshair’，‘arrow’(缺省值)，‘watch’，‘topl’，‘topr’，‘botl’，‘botr’，‘circle’，‘cross’，‘fleur’，‘left’，‘right’，‘top’，‘bottom’，‘fullcrosshair’，‘ibeam’和‘custom’
PointerShapeCData	16×16的矩阵表示用户自定义的鼠标指针。只有属性Pointer设为‘custom’时PointerShapeCData才可用。矩阵中1代表黑色，2代表白色，NaN代表透明色
PointerShapeHot-Spot	是一个有两个元素的向量，表示PointerShapeCData中的点(行和列)。这个点规定了鼠标指针的位置，缺省值为[1，1]
Position	位置向量[*left bottom width height*]，表示图形对象在屏幕上的位置和大小
Renderer	决定绘图的方式。有下列值可以选择： ‘painters’ 适合于画简单的图形 ‘zbuffer’当图形复杂时，比用‘painters’的速度快而且精确，但是需要更多的内存 ‘OpenGL’比‘painters’和‘zbuffer’更高级，但是它需要软件或者硬件的支持。
RendererMode	作图使用的方法。如果 RendererMode设为‘auto’(缺省值)，MATLAB就会根据图形的复杂程度选择一种方法来作图。如果设为‘manual’，MATLAB就不改变作图方法。
Units	MATLAB中位置和大小的度量单位，有下列值可供选择： ‘pixels’ (标准)屏幕像素 ‘normalized’ 归一化坐标 ‘inches’ 英寸 ‘centimeters’ 厘米 ‘points’ 排字机的点，等于0.353毫米 ‘characters’ 字符 单位的选择会影响属性CurrentPoint和Position
WindowStyle	图形窗口模式，值可以为‘ormal’(缺省值)或者‘modal’。后一种情况下所有的输入(鼠标的点击和通过键盘输入的内容)都被限制在窗口内，直到将它设为‘normal’或者将Visible设为‘off’。模式窗口不显示任何菜单或者用户子菜单。MATLAB中可以用Control—C将所有的模式窗口转换成普通窗口
Clipping	不起作用
Parent	图形父对象的句柄，总是0
SelectionHighLight	不起作用

和图形对象有关的事件用下列的属性和方法来处理。

表14-7 图形对象的属性和方法驱动的事件

CloseRequestFcn	图形窗口关闭时执行的函数(缺省是‘closereq’)
CurrentCharacter	键盘上最新按下的字符键
CurrentAxes	当前坐标轴的句柄，也可参见命令集155中命令gca
CurrentObject	当前对象的句柄，也就是最近被选择的对象，见下面的 CurrentPoint。也可参见命令集155中命令gco

(续)

CurrentPoint	鼠标最近一次按下时所在的位置向量
KeyPressFcn	当鼠标指针在图形内，按下键时执行的回调函数，也就是脚本文件或者MATLAB命令
Resize	是否允许用户使用鼠标重新设定图形窗口的尺寸。它的值可以为‘ on’(缺省值)或者‘ off’
ResizeFcn	当图形窗口重新设定尺寸时运行的函数
SelectionType	鼠标最新的选择类型。UNIX下有如下的选择:‘ normal’(左键，缺省值)，‘ extended’(shift—左键或者中间键)，‘ alternate’(control—左键或者右键)和‘ open’(双击)
WindowButtonDownFcn	当在图形窗口内鼠标按下一个键时运行的函数
WindowButton—MotionFcn	当鼠标指针移动到图形窗口时运行的函数
WindowButtonUpFcn	当在图形窗口内释放开鼠标一个键时运行的函数

下面的属性和方法是用来控制图形对象及其子对象颜色的。

表14-8 图形对象及其子对象的颜色属性和方法

Color	图形窗口的背景色，一个RGB三元组或者MATLAB预定义的颜色名
Colormap	$m\times3$的矩阵。包含m个不同的RGB三元组(颜色)来制成供多边形、曲面和图象对象使用的颜色表。矩阵缺省包含64种颜色，也可参见13.6节
Dithermap	$m\times3$的矩阵(颜色表)。当在‘ 伪彩’(8位或者更低)屏幕上查看‘ 真彩’数据时使用的颜色表。缺省的64色颜色表也可用
DithermapMode	当在‘ 伪彩’(8位或者更低)屏幕上查看‘ 真彩’数据时，决定是人工还是自动转换颜色。如果设置为‘ manual’(缺省值)，则颜色表Dithermap可用(见上)。如果设置为‘ auto’，MATLAB会基于当前的颜色生成一个颜色表。‘ auto’模式产生的效果要比‘ manual’模式好，但是花费的时间要多一些
FixedColors	$m\times3$的只读矩阵(颜色表)，Colormap没有给出颜色，如线条、文本和用户控制对象的颜色
MinColorMap	MATLAB存储到颜色表 Colormap中最少的系统颜色数。应该等于Colormap中的行数，缺省值为64
ShareColors	MATLAB保存Colormap的方式，值可以为‘ on’(缺省值)或者‘ off’。想要Colormap能快速改变，最好用‘ off’。见helpdesk可得更多信息

下列属性用来控制图形对象的输出打印。

表14-9 图形对象的输出属性

InvertHardCopy	是否互换图形对象的打印颜色。缺省值是‘ on’，即图形的背景色为白色，而对象的颜色为黑色。‘ off’正好和它相反
PaperUnits	MATLAB输出图形时使用的度量单位。有下列单位可用:‘ normalized’，‘ inches’(缺省值)，‘ centimeters’和‘ points’。度量单位的选择会影响属性PaperSize和PaperPosition
PaperOrientation	打印图形时纸张的方向。值可以为‘ portrait’(垂直方向，缺省值)或者‘ landscape’(水平方向)
PaperPosition	位置向量[*left bottom width height*]，代表页面上图形打印的位置



(续)

PaperPositionMode	图形输出在纸张上的位置是手动还是自动给出。如果PaperPositionMode设为‘manual’(缺省值)则MATLAB会使用PaperPosition(见上)，如果PaperPositionMode设为‘auto’，输出的位置就和在屏幕上看到的一样。
PaperSize	有两个元素的只读向量[*width height*]，规定打印纸张的大小。
PaperType	打印图形纸张的类型，有下列值可供选择：‘usletter’(缺省)、‘uslegal’、‘A0’、‘A1’、‘A2’、‘A3’、‘A4’、‘A5’、‘B0’、‘B1’、‘B2’、‘B3’、‘B4’、‘B5’、’arch—A’、‘arch—B’、‘arch—C’、‘arch—D’、‘arc—E’、‘A’、‘B’、‘C’、‘D’、‘E’和‘tabloid’

■ **例**14.7

(a) 用给定的名字取代窗口的数字标题，可以用如下命令：

```
fp = figure;
set(fp,'NumberTitle','off');
set(fp,'Name','ExampleWindow');
```

(b) 改变窗口的位置和大小：

```
set(fp,'Position',[100 100 400 400]);
```

■

14.2.5 轴对象

轴对象可以在图形窗口中定义画图区域，所以它的父对象是图形对象，它的子对象是线条、图象、补片、曲面和文本对象。

轴对象的属性不仅可以定义位置，还可以规定画图区域内的图形方向和图形大小。

命令集158 轴对象

axes(*prstr*,*alt*,···)	创建轴对象，根据缺省的属性值创建覆盖全部窗口的坐标轴。可选属性**prstr**的值设为*alt*，在它们之后还可以设置几个属性及其相应的值。返回得到坐标轴的句柄。
axes(h)	设置当前的轴对象为*h*。

因为可以定义轴对象的大小，所以在同一个图形对象中可以有多个轴对象；和 subplot命令相比。图形命令只作用在当前轴对象上。

轴对象有下列属性和方法。

表14-10 轴对象的常用属性和方法

Box	坐标轴是否有边框，值可以为‘on’或者‘off’(缺省值)。
CurrentPoint	包含最新按下鼠标位置的点的坐标轴对象上的 2×3的矩阵$\begin{pmatrix} x_b & y_b & z_b \\ x_f & y_f & z_f \end{pmatrix}$，它定义了从坐标空间前面延伸到后面的一条三维直线。
DataAspectRatio	向量[*dx dy dz*]定义图形数据在*x*、*y*和*z*方向上各自的分量。
DataAspectRatioMode	MATLAB是否自动计算数据在 *x*、*y*和*z*方向上各自的分量。如果DataAspectRatioMode设为‘auto’(缺省值)，MATLAB会产生DataAspectRatio；如果设为‘manual’，就直接使用DataAspectRatio。

(续)

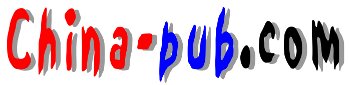
	如果DataAspectRatio设定，那么DataAspectRatioMode就会自动地设为'manual'
DrawMode	画图象使用的模式，值可以为'normal'(缺省值)或者'fast'。后者更快，但会产生不正确的图。当父对象(图形对象)用Renderer命令设置为'zbuffer'时，该属性无意义
LineStyleOrder	指定线条和点的类型的字符串，用于在坐标轴上画多条线。例如：'—*| :|o'将通过点划线、点线和实线循环。LineStyleOrder缺省值为'-'，即所有的线是实线
LineWidth	X、Y和Z坐标轴的宽度，缺省值为0.5点
NextPlot	指定在坐标轴内画新图形的方式： 'add'　(缺省值)使用当前的坐标轴 'replace'　删除窗口及其子对象，并重新设置所有属性值(除位置属性外)。等同于命令cla reset 'replace children'　删除所有子对象，但是不重新设置属性值。等同于命令cla
PlotBoxAspectRatio	向量[*px py pz*]分别规定了由*Lim(见表14-13)定义的坐标框架在*x*，*y*和*z*方向的大小
PlotBoxAspect-RatioMode	MATLAB是否自动计算坐标边框在*x*，*y*和*z*方向的分量。如果设为缺省值'auto'，MATLAB会生成PlotBoxAspectRatio；如果设为'manual'，就直接使用PlotBoxAspectRatio。如果设置了PlotBoxAspectRatio，那么PlotBoxAspectRatioMode就自动地被设为'manual'
Projection	观察模式的设定。Projection设为缺省值'orthographic'，则将数据集合内的平行线平行地画在屏幕上。这种模式适合于数值数据。另一种模式是'perspective'，从远处看对象在屏幕上变得更小，所以这种模式适合于'real'对象，例如实心模型
Position	位置向量[*left bottom width height*]代表坐标轴在图形窗口中的位置和大小
Title	坐标轴标题文本对象的句柄
Units	大小和位置属性值的度量单位。有下列单位可供选择：'inches'、'centimeters'、'normalized'(缺省值)、'points'、'pixels'和'characters'。单位的选择会影响Position属性，见上
View	这是一个旧属性，用CameraPosition、CameraUpVector和Camera-ViewAngle代替，见表14-12，也可参见命令view

■ **例**14.8

假设要建一个用带有限参数的subplot命令不能显示出的图像。用axes命令创建五个坐标轴。

```
[X,Y]  = meshgrid(-2:0.3:2,-2:0.3:2);
ZS     = cos(X).*cos(Y);
                                         % 坐标轴位置
a(1) = axes('Position',[0.1 0.1 0.2 0.2]); % 左下
a(2) = axes('Position',[0.8 0.1 0.2 0.2]); % 右下
a(3) = axes('Position',[0.8 0.8 0.2 0.2]); % 右上
a(4) = axes('Position',[0.1 0.8 0.2 0.2]); % 左上
a(5) = axes('Position',[0.3 0.3 0.5 0.5]); % 中间

for i = 1:5,
  axes(a(i));                  % 在不同坐标轴内画图
```




```
    surf(ZS);

    if i == 1                    % 在不同的坐标轴内改变观察点
      view(37.5,30);
    elseif i == 2                % 中间坐标轴使用的是缺省角度值(-37.5, 30)
      view(-37.5,70);
    elseif i == 3
      view(10,30);
    elseif i == 4
      view(0,-20);
    end;
  end;
```

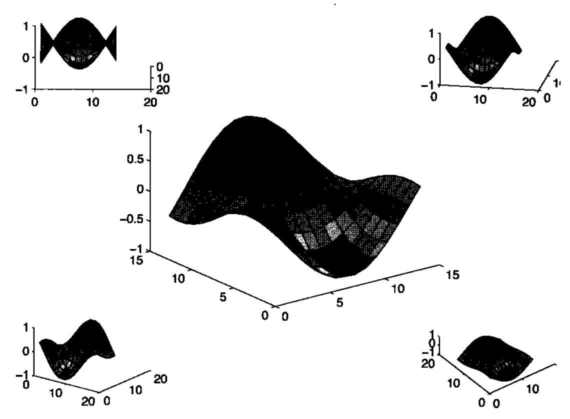


图14-8 用axes 画的不同坐标轴位置

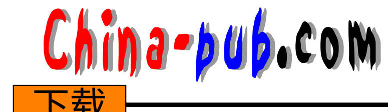
注意，前两个数定义坐标轴的左下角位置，后两个数定义坐标轴的宽度和高度。在当前的窗口内将位置归一化到[0，1]之间，结果如图14-8所示。这是从不同的角度看同一个曲面在坐标轴内的图形。

下列属性用来控制轴对象和它子对象的颜色。

■

表14-11 控制轴对象和它子对象颜色的属性和方法

AmbientLightColor	是一个RGB三元组或者是一种MATLAB中预定义的颜色，用来指定背景光的颜色。它用相同的数量影响所有对象。仅在轴对象中有一个光源对象时，它才有效
CLim	颜色界限向量[*cmin cmax*]，它确定将多边形和曲面对象的颜色数据(CData)映射到图形对象的颜色表Colormap。*cmin*是映射到颜色表中的第一种颜色，*cmax*映射到最后一种颜色。CData中的值线性映射到在*cmin*和*cmax*之间的值，小于*cmin*的值和大于*cmax*的值分别映射到第一种或者最后一种颜色上
CLimMode	MATLAB是否自动计算CLim。如果CLimMode设为缺省值‘auto’，则MATLAB在所有可见的图形对象中搜索出CData的最小和最大值来。如果CLimMode设为缺省值‘manual’，就直接使用CLim。设置CLim就把CLimMode设为‘manual’
Color	坐标轴背景颜色；缺省值‘none’表示坐标轴是透明的。颜色可以是一个RGB三元组或者一个MATLAB中预定义的颜色
ColorOrder	一个$m \times 3$RGB值的矩阵，用在plot和plot3中。如果用户没有指定颜色，这些函数就从这个矩阵中循环地使用颜色



MATLAB中使用‘camera’来指定观察坐标轴中的对象的角度。下列属性用来控制‘camera’。

表14-12 轴对象的观察点属性

CameraPosition	位置向量[*x y z*]，规定在坐标轴内观察点的位置
CameraPositionMode	决定MATLAB是否自动计算观察点位置。如果CameraPositionMode设为缺省值‘auto’，MATLAB会生成CameraPositionMode。如果设为‘manual’，就直接使用CameraPositionMode。设置CameraPosition，就自动把CameraPositionMode设为‘manual’
CameraTarget	位置向量[*x y z*]，规定在坐标轴内目标观察点的位置
CameraTargetMode	决定MATLAB是否会自动计算目标观察点的位置。如果CameraTargetMode设为缺省值‘auto’，MATLAB就将CameraTarget值设为轴对象的中心点；如果设为‘manual’，就直接使用CameraTarget。如设置CameraTarget，就自动把CameraTargetMode设为‘manual’
CameraUpVector	坐标轴内的位置向量[*x y z*]，规定了观察点的旋转角度。在三维视图中缺省值为[0 0 1] 这表示*z*正半轴上的点是向上的
CameraUpVectorMode	决定MATLAB是否会自动计算CameraUpVector。如果CameraUpVectorMode设为缺省值‘auto’，那么在三维视图中将CameraUpVector设为[0 0 1]。在二维视图中将CameraUpVector设为[0 1 0]。如果CameraUpVectorMode设为‘manual’，就直接使用CameraUpVector。设置CameraUpVector，就自动把CameraUpVectorMode设为‘manual’。
CameraViewAngle	观察点的角度(0～180度)，观察角度越大，对象看起来就越小
CameraViewAngleMode	决定MATLAB是否会自动计算观察点角度。如果CameraViewAngleMode设为缺省值‘auto’，MATLAB就会将CameraViewAngleMode设为能观察到所有对象的可能最小角度；如果设为‘manual’，就不改变设置。设置CameraViewAngle，就自动把CameraViewAngleMode设为‘manual’

轴对象和它子对象的轴、刻度、图注和格栅由下列属性控制，其中字符*代表X、Y和Z之一。

表14-13 控制轴的轴对象属性和方法

GridLineStyle	轴对象中格栅使用的线型，可选值有‘—’,‘——’,‘ :’(缺省值)，‘—.’和‘none’。和命令集132中的命令grid比较
Layer	坐标轴放置在轴的子对象上的位置。子对象必须是二维视图，或者View值为[0 90]和DrawMode值为‘fast’的三维视图，见表14-10。Layer的值可以为‘bottom’(缺省值)或者‘top’，也就是分别将坐标轴放在画图区域的下面或上面
TickLength	向量[2*Dlen* 3*Dlen*]，分别代表了在二维和三维视图中的坐标轴刻度标记的长度。TickLength的值是坐标轴的最长长度分之一
TickDir	坐标轴刻度标记的指向。二维视图的缺省值是‘on’，表示刻度标记从坐标轴线向内；三维视图的缺省值是‘out’，表示刻度标记从坐标轴线向外
TickDirMode	MATLAB是否自动设置TickDir。如果TickDirMode设为缺省值‘auto’，就设置TickDir；如果设为‘manual’，就直接使用TickDir。设置TickDir，就自动把TickDirMode设为‘manual’
XAxisLocation	刻度和数字文本放在*x*轴上的位置，值可以为‘bottom’(缺省值)或者‘top’，表示将刻度和文本放在图的下面或者图的上面
YAxisLocation	刻度和文本(数字)放在*y*轴上的位置，值可以为‘left’(缺省值)或者‘right’，表示将刻度和文本放在图的左边或者图的右边



(续)

*Color	一个RGB三元组或者MATLAB中一种预定义的颜色，表示*轴的颜色。在*方向的刻度标记、数字文本和格栅线都是这种颜色，缺省为‘white’(白色)
*Dir	表示*轴的方向，值可以为‘normal’(缺省值)或者颠倒方向的‘reverse’
*Grid	是否在*方向画格栅线，值可以为‘on’或者‘off’(缺省值)。使用命令grid来画或清除在各方向上的格栅线，见命令集132
*Label	*轴标志文本对象的句柄
*Lim	向量[*min max*]，表示*轴的最小和最大值
*LimMode	MATLAB是否自动计算*Lim。如果*LimMode设为缺省值‘auto’，就自动计算*Lim；如果设为‘manual’，就直接使用*Lim。设置*Lim，就自动把*LimMode设为‘manual’
*Scale	表示*轴的线性换算或者对数换算，值可以为‘linear’或者‘log’
*Tick	是一个元素值单调递增的向量，表示刻度标记画在*轴上的位置
*TickLabel	数字文本字符串矩阵，用在*轴上标出刻度标记。如果矩阵中字符串不够，就重复使用
*TickLabelMode	MATLAB是否自动计算*TickLabel。如果*TickLabelMode设为缺省值‘auto’，就设置*TickLabel；如果设为‘manual’，就直接使用*TickLabel。设置*TickLabel，就自动把*TickLabelMode设为‘manual’
*TickMode	MATLAB是否自动计算*Tick。如果*TickMode设为缺省值‘auto’，就设置*Tick；如果设为‘manual’，就直接使用*Tick。设置*Tick，就自动把*TickMode设为‘manual’

轴对象中坐标轴上的文本和数字显示由下列属性控制。

表14-14　控制坐标轴文本的属性

FontAngle	字符串表示坐标轴文本字体的角度，设定值可以为：‘normal’(缺省值)、‘italic’和‘oblique’
FontName	字符串表示使用的字体
FontSize	字体的大小，使用FontUnits中的单位，见下面。
FontUnits	FontSize中使用的字体大小单位，可以用下列选择：‘inches’、‘centimeters’、‘normalized’、‘points’(缺省值)和‘pixels’
FontWeight	坐标轴文本加黑，设置值可以为：‘light’、‘normal’(缺省值)、‘demi’和‘bold’

坐标轴的区间可以用给定数据的最大值和最小值，每个刻度之间单位可以用缺省值 1、2或者5。可以通过属性‘XTick’和‘YTick’对刻度和格栅线进行操作。

■ **例**14.9

下面的程序给出如何来改变坐标轴刻度标记和格栅线。

```
x = [1 3 7];                % 生成X向量
y = [6 9 2];                % 和y向量

s1 = subplot(2,2,1);        % 左上角建一个坐标系
plot(x,y);
grid;                       % 使用缺省的格栅线
title('Default');
```

```
s2 = subplot(2,2,2);                % 右上角建一个坐标系
plot(x,y);
set(s2,'XTick',x);                  % 改变X轴标记
set(s2,'XGrid','on');               % 画X轴的格栅线
title('X scale manipulated');

s3 = subplot(2,2,3);                % 左下角建一个坐标系
plot(x,y);
set(s3,'YTick',[2 6 9]);            % 改变y轴标记
set(s3,'YGrid','on');               % 画y轴的格栅线
set(s3,'GridLineStyle','-.');       % 使用虚线格栅
title('Y scale manipulated');

s4 = subplot(2,2,4);                % 右下角建一个坐标系
plot(x,y);
set(s4,'XTick',x);                  % 改变xy轴标记
set(s4,'YTick',[2 6 9]);
grid;                               % 画xy轴的格栅线
title('Both scales manipulated');
```

得到的结果如图14-9所示。

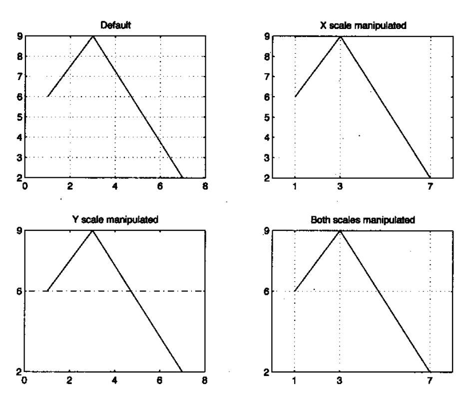


图14-9 不同的刻度标记和格栅的坐标轴

■

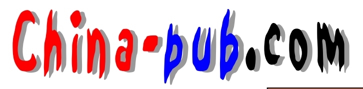
14.2.6 用户控制对象

用户控制对象用来建立图形用户界面，即 **GUI**。它们可以用来创建在 **GUI**中的所有控制对象，比如，按钮、滑标和无线按钮等。正是因为此，在程序中使用用户控制对象使得程序处理起来更容易。

有关GUI的更多信息可参见14.3节。

命令集159 用户控制对象

uicontrol(*f p*, *prstr*,val,...)	在句柄为*f p*的图形窗口中创建控制窗口，返回得到这个控制窗口的句柄。如果没有给出 *f p*，则在当前图形窗口中创建。用 **prstr**和*alt*来设置属性及其相应的值：类型(按钮、滑标等)、大小、位置等等。可以同时对多个属



	性进行设置。属性‘ CallBack ’控制用户选择结果。
[outstr,pos]= textwrap(h,instr)	将文本粘贴在**instr**句柄*h*的用户控制对象上。参数**instr**是一个细胞向量，每个细胞是一行文本内容，**outstr**是粘贴的文本。向量**pos**包含的是用户控制对象的缺省位置和大小。
popupstr(h)	句柄*h*的弹出式菜单中当前被选中的项(文本)。弹出式菜单是将属性Style设为‘popupmenu’的一个用户控制对象，见下面。

用户控制对象有下列属性和方法。

表14-15　用户控制对象的属性和方法

BackgroundColor	是一个RGB三元组或MATLAB一个预定义的颜色，来设置对象的背景色
Callback	当用户激活用户控制对象如点击对象时运行的回调函数。‘ frame ’和‘ text ’类型的用户控制对象的回调函数不可交互选择
CData	是一个 RGB值的 $m \times n \times 3$ 矩阵，表示在‘ pushbutton ’或‘ togglebutton ’上显示的真彩图像
Enable	用鼠标点击用户控制对象时用户对象的动作，有下列值可以选择： ‘ on ’　(缺省值)表示执行‘ Callback ’ ‘ inactive ’　表示执行‘ ButtonDownFcn ’ ‘ off ’　表示执行‘ ButtonDownFcn ’，同时对象的标题 (见String)变得模糊不清
Extent	只读向量[0 0 *width height*]表示文本String的大小
ForegroundColor	一个RGB三元组或MATLAB一个预定义的颜色，设置文本String的颜色，见下面。缺省的文本颜色为黑色
Horizontal Alignment	标志文本String的水平排列。值可以为：‘ left ’、‘ center ‘ (缺省值)和 ’right ’。
ListboxTop	用在‘ listbox ’对象中的一个数值，表示将细胞向量 String中以这个数值为下标的字符串放在列表框的最上面
Max	表示属性Value(见下面)的最大值，取决于控制对象的类型： • 当‘ radiobutton ’、‘ checkbox ’和‘ togglebutton ’处于‘ on ’状态时，将它们的Value设定为Max • 对于‘ slider ’，它就是可选的最大值 • 对于‘ edit ’Max，当Min>1时，可编辑文本框是多行文本 • 对于‘ listbox ’ Max，当Min>1时，表示列表框中有多个选项可以选择 • 对于对象‘ frame ’、‘ popupmenu ’和‘ text ’，它是无效属性
Min	表示属性Value(见下)的最小值，取决于控制对象的类型： • 当‘ radiobutton ’、‘ checkbox ’和‘ togglebutton ’处于‘ off ’状态时，将它们的Value设定为Min • 对于‘ slider ’它就是可选的最小值 • 对于‘ edit ’Max，当Min>1时，可编辑文本框是多行文本 • 对于‘ listbox ’Max，当Min>1时，表示列表框中有多个选项可以选择 • 对于对象‘ frame ’、‘ popupmenu ’和‘ text ’，它是无效属性。
Position	位置向量[*left bottom width height*]表示对象在屏幕上的位置

(续)

String	用来设置对象文本的字符串。对于‘popupmenu’来说，如果String是一个细胞向量，字符串矩阵或者是用垂直条‘ | ’分隔用括号括起来的字符串如(‘popupmenu’,‘listbox’)或‘\n’(‘edit’,’text’)。可以用这个字符串来设置‘listbox’、‘edit’和‘text’的几个参数和文本的线条
Style	定义对象类型的字符串。有下列值可以选择：‘pushbotton’(缺省值)、‘radiobutton’、‘checkbox’、‘edit’(可编辑文本)、‘text’、‘slider’、‘frame’、‘listbox’、‘popupmenu’和‘togglebutton’(开/关按钮)
SliderStep	向量*[minstep maxstep]*分别表示每次单击滑标箭头移动的最小范围和单击滑标移动的最大范围值。缺省值为[0.01 0.1]
TooltipString	用户将鼠标移动到控制对象上显示的提示字符串
Units	位置属性值的单位，有下列单位可以选用：‘inches’、‘centimeters’、‘normalized’、‘points’、‘pixels’(缺省值)和‘characters’。单位的选择会影响属性Extent和Position，见上面。
Value	不同类型对象的值是不同的 • 对于‘radiobutton’、‘checkbox’和‘togglebutton’分别参见上面的Max和Min属性 • 对于‘slider’，它就是当前值 • 对于‘popupmen’，它是当前可选项的个数 • 对于‘listbox’，它是包含当前可选项的向量 • 对于其他的用户控制对象，这个属性是无效的
Children	用户对象没有子对象，总是空矩阵[]
Clipping	无效属性
HitTest	无效属性

用户控制对象中的文本显示由下列属性控制。

表14-16 用户控制对象中的文本控制属性

FontAngle	FontName	FontSize	FontUnits	FontWeight

在表14-13中对它们进行了详细描述。

14.2.7 用户菜单对象

用户菜单对象和用户控制对象一样也是用来建立图形用户界面的。可以用它来建立下拉式菜单，将它放置在图形窗口的顶部。

命令集160 用户菜单对象

uimenu(*p,prstr,alt*,...)	在句柄*p*的图形窗口顶部建立一个下拉式菜单，返回这个对象的句柄。如果没有指定*p*，就使用当前窗口句柄。如果句柄*p*已经和一个菜单相关联，则建立子菜单，所以可以建立多级菜单。菜单的属性和它的名字等可以由参数**prstr**和*alt*给出，将属性**prstr**设置为值*alt*。可以对多个属性及其属性值进行设置。



makemenu(h,mencho, calls,*tags*)	在句柄*h*的图形对象中创建一个菜单结构。参数 **mencho**是包含菜单选项的字符串矩阵， **calls**是一个字符串矩阵，包含的是当用户在菜单中选择菜单选项时 MATLAB要执行的命令。如果给出字符串矩阵 **tags**，就设置作为菜单一部分的菜单对象的属性 Tag(见表14-2)。命令makemenu返回创建对象的句柄向量，用help makemenu可以得到更多信息。也可以用工具menuedit来建立菜单，参见14.3.5节。
umtoggle(h)	句柄*h*的用户菜单对象的选择状态。如果选中用户菜单对象，则返回 1 ；否则返回 0。也可参见下面的用户菜单属性Checked。
winmenu(*h*)	创建句柄*h*的图形窗口中的菜单 Window的子菜单。如果没有给出*h*，则用当前图形窗口的句柄。菜单 Window的Tag属性必须设置为‘ winmenu ’，见表14-2。用help winmenu可以得到更多信息。

用户菜单对象有下列属性和方法。

表14-17　**用户菜单对象的属性和方法**

Accelerator	指定菜单项的快捷键，用户可以用Control—Accelerator来选择菜单项；Macintosh系统，用Command—Accelerator来选择。这对有回调函数但是没有子菜单的选项才有效
Callback	表示当用户选中菜单项时运行的回调函数
Checked	菜单项的选中状态，值可以为‘ on ’或者‘ off ’(缺省值)
Enable	菜单使能状态，值可以为‘ on ’(缺省值)或者‘ off ’。当设为‘ off ’时，菜单项将变成灰色
ForegroundColor	用户菜单的前景(文本)色，是一个RGB三元组或者MATLAB预定义的颜色。注意：这个属性只能使用在X Windows系统中
Label	含有菜单项名字的文本串，用‘ & ’放在名字前会将名字中的一个字符加上下划线。这样就可以用键盘上相对应于带下划线的字符的键来激活菜单项
Position	表示菜单行中的相对位置，值1表示最左边，在菜单中表示最上面
Separator	分割符，表示在菜单项上是否画一个分割线。值为‘ on ’或者‘ off ’(缺省值)
ButtonDownFcn	无效属性
Children	包含子菜单的句柄向量
Clipping	无效属性
Selected	无效属性
SelectionHightLight	无效属性
UIContextMenu	无效属性

14.2.8　用户快捷菜单对象

Uicontextmenu对象可以用建立快捷菜单，这是 MATLAB 5.2中介绍的一种新型菜单。当

鼠标指针停在对象上时，如用户按下鼠标右键 (在Macintosh系统按下Control—单击)就可以显示出这些菜单来。从这个菜单中用户可以进行选择，快捷菜单总是和其他的对象联系在一起，参见表14-2中的属性UIContextMenu。

命令集161 用户快捷菜单对象

uicontextmenu (*prstr*,*alt*, ···)	建立快捷菜单，将属性prstr设置为*alt*，返回菜单的句柄。可以对多个属性进行设置。

Uicontextmenu对象有下列属性和方法。

表14-18 Uicontextmenu对象的属性和方法

Callback	当在对象上按下鼠标右键 (在Macintosh系统中按下Control—单击) 时运行的回调函数
Children	包含用户菜单对象句柄的向量，也就是快捷菜单中的菜单项
ButtonDownFcn	无效属性
Clipping	无效属性
HitTest	无效属性
Selected	无效属性
SelectionHighlight	无效属性
UIContextMenu	无效属性

■ **例14.10**

在这个例子中将建立曲面对象的快捷菜单，用户可以在这个菜单中改变曲面的线型。

```
% 画一个球体曲面
figure(1)
sphere;
sp = findobj(1,'Type','surface');        % 获取曲面指针

% 定义菜单选项，即回调函数。
cb1 = ['set(sp,''LineStyle'',''none'')'];
cb2 = ['set(sp,''LineStyle'',''--'')'];
cb3 = ['set(sp,''LineStyle'','':'')'];
cb4 = ['set(sp,''LineStyle'',''-'')'];

% 定义快捷菜单
cmenu = uicontextmenu;
set(sp,'UIContextMenu',cmenu)
menp  = uimenu(cmenu,'Label','Linetypes');
item1 = uimenu(menp, 'Label','none',  'Callback',cb1);
item2 = uimenu(menp, 'Label','dashed','Callback',cb2);
item3 = uimenu(menp, 'Label','dotted','Callback',cb3);
item4 = uimenu(menp, 'Label','solid', 'Callback',cb4);
```

用户按下鼠标右键(Macintosh系统中为Control—单击)，图14-10中的菜单就会显示出来。

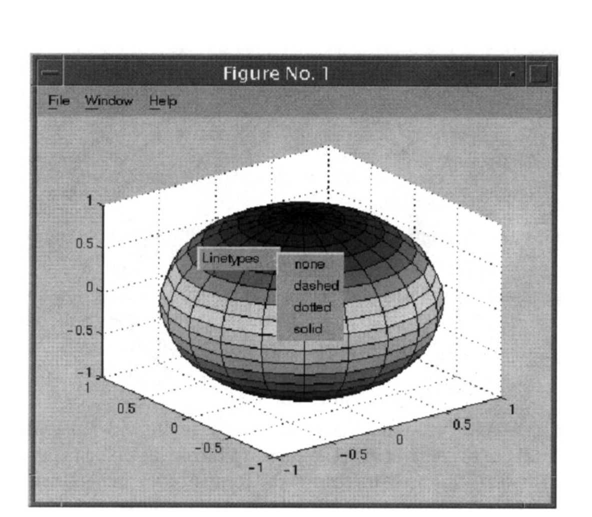


图14-10 有快捷菜单的曲面对象

■

14.2.9 图像对象

MATLAB中也可以显示图像。MATLAB中的图像是由矩阵来定义的，矩阵的元素相对应于图像中的点，元素的值相对应于点的颜色，所以 MATLAB中的图像是‘位图’类型。

命令image既是一个高级命令又是一个低级命令，可以用来在屏幕上画出由矩阵定义的图像。相应的捕获图形窗口图像的函数是命令 capture。然而要注意的是通常命令capture和image给出的结果要比MATLAB内部表示的要低级一些，而且需要较多的内存。

命令集162 图像对象

`image(C)`	把矩阵**C**作为一个图像画出，**C**中的每个元素是图像的一个点，元素的值表示颜色。如果没有给出**C**，就用缺省值。最后返回得到图像的句柄。
`image(x,y,C)`	把矩阵**C**作为一个图像画出，向量**x**和**x**分别表示*x*、*y*轴的最大和最小值。
`image(`*`prstr,alt,`*`...)`	用设定值**prstr**和*alt*来创建一个新图像，可以同时对多个属性进行设定。这些属性可以用 set命令来修改。
`imagesc(...)`	和image相同，但是标定数据以使用全部的颜色表。
`capture(h)`	建立一个与句柄*h*窗口有相同内容的新窗口。
`[C,Cm]=capture`	返回用相应颜色表**Cm**的图像矩阵**C**，但是并不画出图像。
`imfinfo(filename,`*`fmt`*`)`	返回得到关于图像文件信息的结构。字符串**filename**是文件名，**fmt**是文件的扩展名。扩展名可以为‘bmp’、‘hdf’、‘jpg’、‘jpeg’、‘pcx’、‘tif’，‘tiff’或者‘xwd’。见helpdesk可得更多信息。
`[A, Ftab]=imread(filename,`*`fmt,idx`*`)`	从图像文件**filename**中将读出图像到矩阵**A**中，矩阵**Ftab**是图像的颜色表。字符串 **fmt**是图像文件的扩展名，见上

	面。如果没有给出扩展名，MATLAB就试着匹配。如果文件中有多个图像(只对扩展名为‘tiff’和‘hdf’的文件)，标量 *idx* 表示从文件读出的图像的个数。见helpdesk可得更多信息。
imwrite(A,*Ftab*, filename,*fmt*, *prstr*,alt)	同上，不同的是将图像写入到文件中。如果没有文件格式**fmt**，文件必须要有**fmt**作为扩展名。用扩展名‘hdf’、‘jpeg’和‘tiff’可以设定图像文件的某些属性；将属性prstr设为*alt*。见helpdesk可得更多信息。

图P-7中显示的地球图像，就是用image命令绘制的。命令colormap(map)给出了正确的颜色。图像对象有下列属性和方法。

表14-19 图像对象的属性和方法

CData	指定图像矩阵中的元素颜色值。数据格式和补片中顶点的颜色格式相同
CDataMapping	见表14-21
EraseMode	见表14-20
Xdata	是一个大小为1×size(CData, 2)的向量，表示图像*x*的坐标
YData	是一个大小为1×size(CData, 2)的向量，表示图像*y*的坐标
Children	空矩阵[]，图像对象没有子对象

■ **例**14.11

不带参数的image命令生成倒立的图像，如图14-11所示。

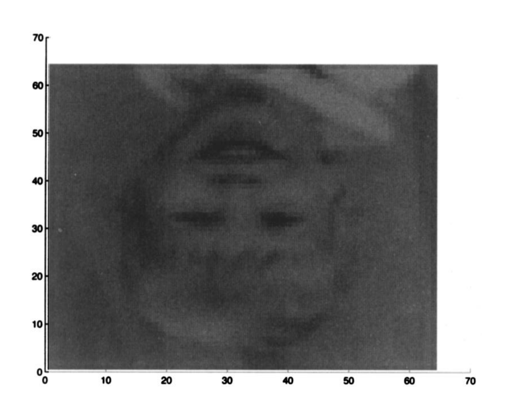
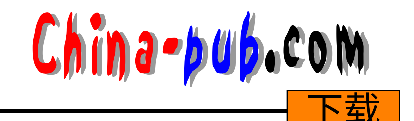


图14-11 不带参数的image 命令生成的图像

■

14.2.10 线条对象

一般不用低级命令line来直接创建线条对象，而是用高级命令，如plot。然而有时也用line命令来操作已有的线条和画新线条。和高级命令plot比较，line命令只是画一条线，而plot命令除了画线条外，还可以做一些其他的事情，如替换坐标轴。



命令集163 线条对象

`line(x,y)`	在当前图形中画向量x和y的直线。如果x和y是同样大小的矩阵，则每列画一条直线。返回得到线条对象的句柄。
`line(x,y,z)`	类似上个命令在三维坐标系中画一条直线。
`line(prstr,alt,...)`	画一条直线，将属性prstr的值设为*alt*。同时可以对多个属性值进行设定。

线条对象有下列属性和方法。

表14-20 线条对象的属性和方法

`Color`	线条颜色，一个RGB三元组或一个MATLAB预定义的颜色	
`EraseMode`	消除和重画模式	
	‘`normal`’	(缺省值)重画影响显示的整个区域，这是最精确的、也是最慢的一种模式
	‘`none`’	当移动或删除线条时该线不会被消除
	‘`xor`’	用图形的背景颜色执行异或(XOR)运算，画出和消除线条。当消除线条时不影响背景色，但是线条颜色取决于下面的图形
	‘`background`’	通过在图形背景色中重画线来消除线条。这样会影响背景色，但是线条颜色取决于线下图形
`LineStyle`	线条类型，有下列可选值：‘—’(缺省值)、‘——’、‘:’、‘—’和‘none’。	
`LineWidth`	以点为单位的线宽，缺省值是0.5点	
`Marker`	标记数据的记号类型，有下列可选值：‘+’、‘o’、‘*’、‘.’、‘x’、‘`square`’、‘`diamond`’、‘^’、‘v’、‘>’、‘<’、‘`pentagram`’、‘`hexagram`’和‘none’(缺省值)	
`MarkerSize`	以点为单位的记号大小，缺省值是6点。注意，MATLAB在指定大小的1/3内标记记号	
`MarkerEdgeColor`	没有填充的点颜色或填充点的边颜色，值可以为一个RGB三元组、一个MATLAB预定义的颜色、‘`none`’或‘`auto`’(缺省值，给出颜色`Color`；见上面)	
`MarkerFaceColor`	填充点的填充色，值可以为一个RGB三元组、一个MATLAB预定义的颜色、‘`none`’或‘`auto`’(缺省值，给出颜色`Color`；见上面)	
`XData`	*x*轴坐标的向量	
`YData`	*y*轴坐标的向量	
`ZData`	*z*轴坐标的向量	
`Children`	空矩阵[]，线条对象没有子对象	

14.2.11 补片对象

补片对象是一个填充的多边形定义的区域。根据维数，多边形的角由两个或三个向量x、y和z定义，并且按向量元素的顺序来定义。多边形用`patch`命令和指定的颜色进行填充。

命令集164 补片对象

`patch(x y c)`	在当前坐标系内画出向量x和y定义的多边形。参数c是多边形的颜色，可以是数值、向量或矩阵，见CData和下面的FaceVertexCData。

	如果**x**和**y**是矩阵，则每列画一个多边形。命令返回补片对象的句柄。
patch(x,y,z,c)	在三维坐标系内画出多边形，和在二维坐标系内相似。
patch(*prstr*,*alt*,...)	创建一个多边形，将它的属性 **prstr**的值设为*alt*，可以同时对多个属性进行设定。

■ **例**14.12

下面是球体和球体补片对象的例子。

```
sphere(10);                % 画一个有121个角的球体
x=[-2-2 2 2];              % 定义多边形的x轴位置
y=[-2 2 2-2];              % 定义多边形的y轴位置
z=[-2-2-2 2];              % 定义多边形的z轴位置
c=[-2-1 1 2] ;             % 定义多边形颜色
p1=patch(x, y, z, c);% 画多边形，并保存其句柄在p1中
```

结果如图14-12所示。

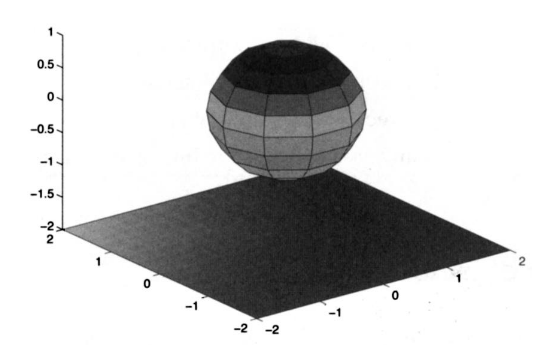


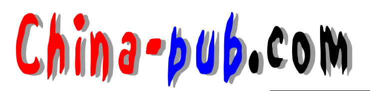
图14-12 用patch 命令画的球体和多边形

■

补片对象的外观和属性可以用set和get来重新设定和获取。补片对象有下列属性和方法。

表14-21 **补片对象的属性和方法**

Cdata	多边形的颜色，也可参见CData Mapping • 一个RGB三元组，MATLAB预定义的颜色之一或是父对象的颜色表中的下标值 • RGB值的$m \times 3$矩阵或给出多边形每个面的颜色值的向量 • RGB值的$m \times n \times 3$矩阵或给出多边形每个角的颜色值的矩阵 在伪色屏幕中需要抖动 RGB三元组，这有利于父对象图形的属性Colormap和Dithermap，见表14-8。如果CData含有NaN，那么MATLAB将不对多边形表面着色，也可参见 FaceVertexCData。
CDataMapping	Cdata或FaceVertexCData中‘color index’(颜色下标)可能的情况。下标是整数，则直接或间接表示父图形对象的颜色表ColorMap(见表14-18)中的一个位置： ‘scaled’(缺省值) MATLAB线性地将颜色数据转换成颜色填充父图形对象中的Clim表示的间隔



(续)

	‘direct’下标直接表示父图形对象的颜色表ColorMap中的颜色。超出颜色表的下标映射到第一个下标(如果值小于1)或者最后一个下标
FaceVertexCData	显示多边形表面或者顶点颜色 • 一个RGB三元组或一个数值表示多边形取得一种颜色，这个数值表示父对象的颜色表的下标(见CDataMapping) • 一个$m\times3$矩阵，m是Faces中的行数，元素是RBG三元组或$m\times1$的向量(见CDataMapping)给出多边形的每个顶点的颜色值也可参见 Cdata
EdgeColor	显示多边形边缘的颜色 • 一个RGB三元组或一个MATLAB预定义的颜色，表示所有的边缘线是单一色，缺省值是‘black’(黑色) • ‘none’表示不画边缘线 • ‘flat’表示边缘线颜色和补片顶点的颜色相同 • ‘interp’表示边缘线颜色由补片属性CData或FaceVertexCData的值线性插值得到
EraseMode	见表14-20
FaceColor	多边形表面颜色 • 一个RGB三元组(缺省值) • ‘none’表示不画表面，但画出边缘 • ‘flat’表示表面颜色和第一个顶点颜色相同 • ‘interp’表示表面颜色通过顶点颜色双线性插值得到
Faces	$m\times n$矩阵，表示顶点的连接状态，用n个顶点定义m个曲面。矩阵中的每行表示每个曲面的连接状态，和NaN不同的行中元素的个数表示顶点的个数。见helpdesk可得更多信息
LineStyle	轮廓的线型，见表14-20
LineWidth	轮廓线的宽度，见表14-20
Marker	多边形顶点的记号类型，见表 14-20
MarkerEdgeColor	见表14-20，缺省值‘auto’表示给出颜色EdgeColor，另一个属性值是非文件式值‘flat’
MarkerFaceColor	见表14-20，属性值‘auto’表示表面颜色是父轴对象的颜色 Color，‘none’表示表面颜色是父图形对象的颜色，还有一个非文件式属性‘flat’
MarkerSize	见表14-20
Vertices	$m\times3$矩阵，表示多边形顶点的x、y、z坐标，也可参见Faces
XData	一个向量或者矩阵，表示顶点的x坐标。如果Xdata是一个矩阵，则每列表示多边形中一个单独表面的x坐标。这种情况下，要求XData、YData和ZData有相同的维数
YData	和XData相同，但是是y坐标
ZData	和XData相同，但是是z坐标
Children	空矩阵，补片对象没有子对象

MATLAB可以通过创建光源对象来照明多边形；见表14-26。有下列属性和方法来控制光效果。

表14-22　控制补片对象光效果的属性和方法

FaceLighting	计算多边形表面光效果的算法 ‘none’　(缺省值)没有光效果 ‘flat’　光从同一个方向照射整个表面。比如，这种方法可用来研究平的表面

(续)

	‘gourand’	表示光效果是对整个表面进行插值得到的。比如，这种方法可用来研究曲面
	‘phong’	和‘gourand’相似，可给出更好的效果，但是需要花费较多时间
EdgeLighting	和FaceLighting相似，但是用于多边形的顶点	
BackFaceLighting	当将远离观察点的顶点表面归一化时，多边形的表面光效果；见表 14-12	
	‘unlit’	表面没有光照
	‘lit’	用通常方式照射表面
	‘reverse lit’(缺省值)	照射表面，就象表面的法线指向观察点。这对分清对象的内部和外部表面很有用
AmbientStrength	0～1之间的数值，表示多边形背景光照的强度 (也可称为周围光)。也可参见轴对象属性AmbientColor	
DiffuseStrength	0～1之间的数值，表示从多边形上漫射光的强度。阴暗对象的属性值较大	
SpecularStrength	0～1之间的数值，表示从多边形上反射光的强度。闪耀对象的属性值较大	
SpecularExponent	≥1的数值，表示多边形的‘mirror-likeness’。数值通常在1～500之间，大多数情况值在5～20之间	
SpecularColor-Reflectance	0～1之间的数值，表示光对象和多边形的颜色对‘镜’光颜色的影响程度，数值0表示都受二者影响；数值 1表示只受光对象影响。也可参见轴对象属性AmbientLightColor；表14-11	
VertexNormals	m×3矩阵，包含多边形m个顶点的曲面法线。矩阵可以用来帮助计算多边形的光效果，但是也可以由用户来设定给出其他的光效果	
NormalMode	MATLAB是否自动设置VertexNormals。如果NormalMode设为缺省值‘auto’，MATLAB就计算VertexNormals；如果设为‘manual’，则直接使用VertexNormals。用户如果设置VertexNormals，那么NormalMode就自动设为‘manual’	

曲面对象可以在三维视图中创建‘飞毯’。

14.2.12 曲面对象

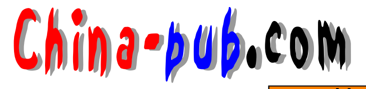
命令集165 曲面对象

surface(x,y,Z,c)	在当前坐标系中创建一个由 **x**、**y**和**Z**定义的填充曲面。向量或者矩阵 **x**和**y**是可选参数，**Z**是一个矩阵，见表 14-23中XData、YData和ZData，参数**c**表示使用颜色的标量或者矩阵，见表14-23中CData。如果没有给出**c**，则令**c**=**Z**。命令返回得到创建的曲面对象的句柄。
surface(*prstr*, *alt*, ...)	创建一个曲面，同时将它的属性 prstr的值设为*alt*，可以对多个属性进行设置。返回得到曲面对象的句柄。

曲面对象有下列属性和方法，其中有一些属性和表 14-21中列出的相同。

表14-23 曲面对象的属性和方法

CData	指定曲面上每个顶点的颜色，也就是ZData中每一点的颜色。如果FaceColor设置为‘texturemap’，那么CData的大小就不必和ZData相同，这样CData中包含的图像被映射到ZData所定义的曲面。见表14-21可知顶点的颜色是如何指定的



(续)

CDataMapping	见表14-21
EdgeColor	见表14-20
EraseMode	见表14-21
FaceColor	见表14-21，而且属性值可以设为‘ texturemap ’，见CData
LineStyle	见表14-21
LineWidth	见表14-21
Marker	见表14-21
MarkerEdgeColor	见表14-21
MarkerFaceColor	见表14-21
MarkerSize	见表14-21
MeshStyle	画网格的边缘行和/或列线：‘ both ’(缺省值)、‘ row ’或‘ colum ’
XData	$m\times n$矩阵或$m\times 1$向量(重复列向量形成$m\times n$矩阵)表示曲面中带内的x坐标
YData	$m\times n$矩阵或$m\times 1$向量(重复列向量形成$m\times n$矩阵)表示曲面中带内的y坐标
ZData	$m\times n$矩阵或$m\times 1$向量(重复列向量形成$m\times n$矩阵)表示曲面中带内的z坐标
Children	空矩阵，曲面对象没有子对象

和多边形一样，曲面对象可以有光照效果，有关的属性和方法总结如下 (表14-24)；详细的描述在表14-22中。

表14-24 控制曲面对象光照效果的属性和方法

FaceLighting	EdgeLighting	BackFaceLighting
AmbientStrength	DiffuseStrength	SpecularStrength
SpecularExponent	SpecularColorReflectance	VertexNormals
NormalMode		

■ **例**14.13

(a) 在例14.12中用下列方法本可以创建了一个球体：

```
[X, Y, Z]=sphere;      % 取球体坐标
ss=surface(X,Y,Z);     % 创建并画出曲面对象
view(3);
```

可以用set命令将对象属性‘ FaceColor ’的属性值‘ flat ’改为‘ interp ’:

```
set(ss,‘FaceColor’,‘interp’)
```

属性‘ FaceColor ’的属性值可以为：

```
set(ss,'FaceColor')
[ none | {flat} | interp | texturemap ] -or-a ColorSpec.
```

属性FaceColor是有效的，例如当给出命令 patch、surface或mesh时，就用‘ texturemap ’来代替‘ interp ’或‘ flat ’。属性值‘ none ’将用在下面的例子中。

(b) 命令mesh是一个用来创建曲面的高级命令，可以得到和用命令 surface一样的创建曲面的句柄。下面的程序用到了命令patch和mesh，并且对对象的属性进行了修改。

```
% 向量X1 Y1和Z1表示三维视图内的一个立方体
```

```
z1 = [1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1];
y1 = [1 -1 -1 1 1 1 1 1 1 -1 -1 1 -1 -1 -1 -1];
x1 = [1 1 1 1 1 -1 -1 1 -1 -1 -1 -1 -1 1 1 -1];

clf                                  % 清除当前图形
p = plot3(x1,y1,z1);                 % 画出立方体的线，并保留句柄

set(p,'LineWidth',3,'Color','b')     % 改变线条属性

[XX,YY,ZZ] = sphere(15);             % 矩阵XX、YY、ZZ定义一个单位球体
hold on;

h1 = mesh(XX,YY,ZZ);                 % 画出单位球体，并保留句柄


% 改变球体属性
set(h1,'EdgeColor','b','FaceColor','c')
h2 = mesh(2.*XX,2.*YY,2.*ZZ);        % 画一个球体(半径为2)，并保留句柄

% Changes sphere properties.
set(h2,'EdgeColor','r','FaceColor','none')

set(gca,'Visible','off')             % 改变当前坐标的可视属性

axis square                          % 使球体园滑
```

结果如图14-13所示。

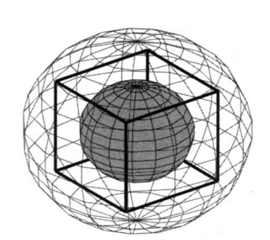


图14-13 大球体中包含一个立方体，立方体中有一个单位球体

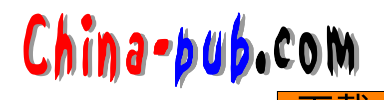
■

14.2.13 文本对象

要将文本显示在图形窗口中，就要用命令 text来创建文本对象。可以指定文本显示的位置，以及改变对象的属性。

命令集166 文本对象

text(x,y,txt)	返回得到文本对象的句柄。在当前二维坐标系中的 (*x*, *y*)位置增加文本字符串**txt**，坐标用当前坐标轴的刻度来指定。
text(x,y,z,txt)	和上个命令一样在当前三维坐标系中增加文本字符串 **txt**。
text(*prstr*, *alt*, ...)	创建一个文本对象，并将其属性 **prstr**的值设置为*alt*。同时可以对多个属性进行设定。



文本对象有下列属性和方法。

表14-25 文本对象的属性和方法

Color	一个RGB三元组或一种MALTAB中预定义的颜色，表示文本的颜色；缺省值为‘white’(白色)，见表14-20
EraseMode	见表14-20
Editing	文本是否可编辑，缺省值‘off’表示文本不可编辑。如果将属性值设为‘on’，在文本的开始处就有一个光标，这样就可以进行文本编辑。完成后需按下ESC键、点击图形窗口或将Editing设为‘off’
Extent	有四个元素的只读向量[*left bottom width height*]表示文本位置(在坐标系中)和文本String的大小
FontAngle	指定字体角度的字符串，可选值为：‘normal’(缺省值)、‘italic’和‘oblique’
FontName	见表14-14
FontSize	见表14-14
FontUnits	见表14-14
FontWeight	见表14-14
Horizontal Alignment	表示文本水平对齐，有以下对齐方式：‘left’(缺省值)、‘center’和‘right’。
Position	向量[*x y*]或[*x y z*]，指出文本在二维或三维空间中的位置
Ratation	以旋转度数表示的文本方向，缺省值为0
String	细胞向量、字符串矩阵或字符串，指定要显示的文本内容
Units	位置属性值的度量单位，有下列度量单位：‘inches’、‘centimeters’、‘normalized’、‘points’、‘pixels’、‘characters’和‘data’(缺省值)。缺省值表示屏幕像素。度量单位的选择影响属性‘Position’和‘Extent’，见上面
Interpreter	String中是否包含LaTeX 文本格式命令(缺省为‘tex’或‘none’)。见helpdesk可得更多信息
VerticalAlignment	表示文本垂直对齐，有以下对齐方式：‘top’、‘cap’(同‘top’但用大写字母)、‘middle’(缺省值)、‘baseline’(字体的基线在指定的y位置)和‘bottom’
Children	空矩阵[]，文本对象没有子对象

■ **例14.14**

在这个例子中，线条和相应的文本颜色相同，另外，改变线条的宽度和文本的字体。

```
clear; clf;

x  = linspace(2,10,100);
y1 = sin(x);
y2 = bessel(1,x);
ha = axes('Position',[0.1 0.1 0.6 0.8]);

l1 = line(x,y1);
t1 = text(x(100),y1(100),'Sine');

l2 = line(x,y2);
t2 = text(x(100),y2(100),'Bessel');
```

```
set(l1,'Color'      ,'Blue'   );          % 设置线1的颜色和线条宽度
set(l1,'LineWidth'  ,3        );

set(t1,'Color'      ,'Blue'   );          % 设置文本的颜色、字体和大小
set(t1,'FontWeight' ,'bold'   );
set(t1,'FontSize'   ,18       );

set(l2,'Color'      ,'Red'    );          % 设置线2的颜色和线条宽度
set(l2,'LineWidth'  ,10       );

set(t2,'Color'      ,'Red'      );        % 设置文本2的颜色、角度、字体和大小
set(t2,'FontAngle'  ,'italic'   );
set(t2,'FontName'   ,'palatino' );
set(t2,'FontSize'   ,30         );

set(ha,'Box','off');                      % 只画出坐标轴，不画出网格
```

结果显示在图14-14中，不幸的是没有显示出颜色来。

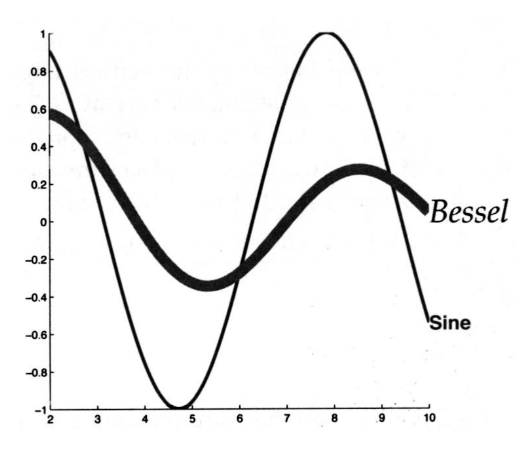


图14-14 有相应文本的线条

■

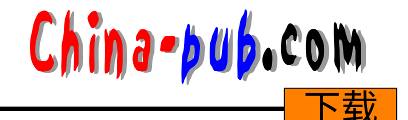
14.2.14 光对象

光对象是MATLAB 5中介绍的一种新类型对象，可以用来创建光源。光源并不真正可见，但是它们能影响补片和曲面的亮度。下列属性均对这些对象的亮度有影响：AmbientStrength、DiffuseStrength、EdgeLighting、FaceLighting、BackFaceLighting、SpecularStrength、SpecularExponent、SpecularColorReflectance和VertexNormals。这些属性的详细描述可见表14-22和表14-24。

可以定义整个坐标轴的背景光，见表 14-11中属性AmbientLightColor。

命令集167 光对象

light(*prstr*, *alt*,…)	创建一个光对象，将其属性 **prstr**的值设置为*alt*，同时可以对多个属性进行设定，返回得到光对象的句柄。



在调色板一节中的图P-2和P-8就是用light命令创建的。使用句柄来对对象的属性进行设定。光对象有下列属性和方法。

表14-26 光对象的属性和方法

Position	向量*[x y z]*表示光在其父对象坐标轴中的位置或方向
Color	一个RGB三元组或一种MATLAB中预定义的颜色，表示光的颜色
Style	光源的类型： 'infinite' (缺省值)表示光源从Position方向发出平行光束 'local' 表示光源的位置在Position，并向所有方向发出光束
ButtonDwnFcn	无效属性
Children	空矩阵[]，光对象没有子对象
Clipping	无效属性
HitTest	无效属性
Interruptible	对回调函数DeleteFcn没有影响
Selected	无效属性
SelectionHighlight	无效属性
UIContextMenu	无效属性

■ **例**14.15

在这个例子中创建一个曲面对象并观察在有光源和没有光源情况下的差别。程序如下：

```
% 用缺省值创建一个曲面对象

subplot(2,1,1)
surf(peaks);
axis('off')
title('unlit surface')

subplot(2,1,2)
sp = surf(peaks);

% 曲面颜色红色，无网格，光照算法'phong'

set(sp,'FaceColor','red')
set(sp,'EdgeColor','none')
set(sp,'FaceLighting','phong')

% 使用平行于X轴的光束照曲面

light('Position',[1 0 0],'Style','infinite')
axis('off')
title('lit surface')
```

程序运行得到的结果如图14-15所示。

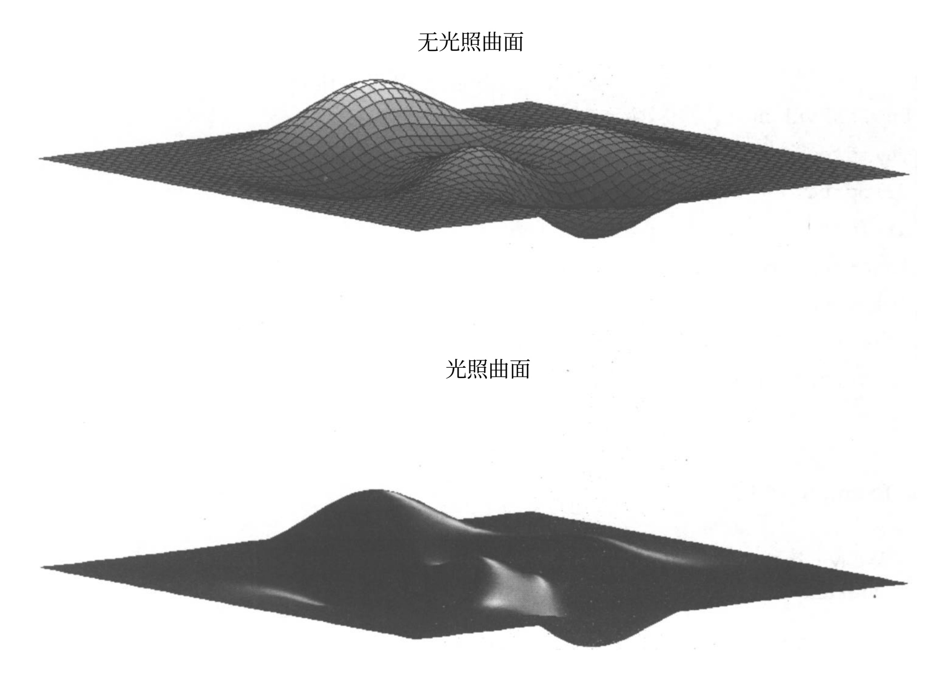


图14-15　有光照和没有光照的曲面

■

14.3　图形用户界面

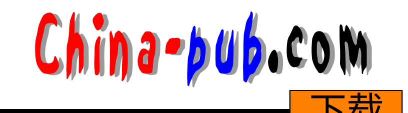
MATLAB有强有力的工具来建立图形用户界面，而且有许多预定义窗口，如下：

命令集168　预定义窗口

`dialog(`*`prstr,alt,···`*`)`	创建一个对话框图形对象，并将其属性**prstr**设置为*alt*，同时可以对多个属性进行设置。窗口是对话框模式，见`msgbox`。返回得到对象的句柄。
`msgbox(message,` *`title,icon,icondata,`* *`iconclr,mode`*`)`	创建一个带有信息的窗口，并返回它的句柄，实际上它也是一个图形对象。 • 参数**message**是一个字符串、字符串矩阵或细胞向量**title**也是一个字符串表示窗口的标题。 • 字符串**icon**表示在窗口上显示的图标类型，有下列几种类型图标：`'none'`、`'error'`、`'help'`、`'warn'`或`'custom'`。 • 只有图标类型是`'custom'`，才能在**icondata**中给出用户自定义图标数据，和使用**iconclr**中的颜色表。 • 参数**mode**表示窗口的模式。有下列几种模式：`'modal'`要求用户反馈信息、`'nonmodal'`表示不必用户反馈信息、`'replace'`表示无对话模式，并替换有相同**标题**的窗体。

helpdlg(*hlpstr*,*title*)	在标题名为**title**的窗体中创建一个有帮助文本 hlpstr的对话框。当用鼠标选定一个按钮，窗体就消失。返回窗体的句柄，也是一个图形对象。
helpwin topic	显示一个MATLAB突出主题topic的帮助窗口，可以是M文件中一个用户自定义命令。在命令帮助中用线条上的‘See Also’将其他相关的命令连接起来。
helpwin(hlpstr, heading,*title*,prstr, *alt*,···)	在标题**heading**的帮助窗口**title**中显示帮助文本。这里的**hlpstr**是帮助文本，可以是细胞向量、字符串矩阵或字符串，每行用一个‘\n’隔开，也就是‘回车’字符。字符串**prstr**是文本对象的一个属性，并将它的属性值设为*alt*，它对帮助文本的显示有影响。可以同时对多个属性进行设置。
helpwin(r1 t1; r2 t2;···,*page*, *title*, *prstr*,*alt*,·)	同上(**r1**、**r2**···是标题，**t1**、**t2**···是帮助文本)，在不同的标题下可能创建出多个帮助页。字符串 **page**必须和其中的一个标题相同，表示开始显示相应的帮助文本，缺省值为r1。
warndlg(*warn*, *title*)	创建一个有警告信息的对话框。文本 **warn**显示在标题为**title**的窗口中。当鼠标选中一个按钮时，窗口就会消失。返回得到窗口的句柄，也是一个图形对象。
errordlg(*errstr*, *title*,‘*on*’)	在标题为**title**的窗口中创建一个带有错误信息的对话框。如果给出‘on’，标题为**title**的窗口存在就将它变成当前窗口；如果窗口不存在，就创建一个标题为 title的新窗口。当用鼠标选中一个按钮时，窗口就会消失。返回得到窗口的句柄，也是一个图形对象。
questdlg(qst, *title*,*alt1*,*alt2*,*alt3*,	创建一个有提问**qst**和标题**title**的对话框。如果可选参数没有给出，窗体上就有三个按钮‘Yes’、‘No’和‘Cancel’，也可以用文本**alt1**、**alt2**和**alt3**给出按钮上的显示文本(至少给出两个)。字符串**default**表示用户按下按钮时返回的内容：如果是使用标准按钮，字符串就等于‘Yes’、‘No’或‘Cancel’；否则等于文本**alt1**、**alt2**或**alt3**。选中一个按钮，窗口就会消失，并且返回按钮上的文本内容。
inputdlg(legend, *title*,*lineNr*,*dfltAns*)	创建一个窗口用来输入数据。细胞向量 **legend**包含答案的图例。字符串**title**表示窗口的名字，**lineNr**是一个包含每个答案行数的向量(或者是总的行数)。细胞向量**dfltAns**包含缺省的答案。命令返回用户给出答案的细胞向量。
menu(title, alt1, alt2,···)	创建一个菜单，它的标题是**title**，有菜单选项**alt1**、**alt2**等，等待从键盘或者鼠标输入。返回用户选择的菜单选项，以整数表示。注意：如果其他图形对象调用

	menu，那么就必须将那个对象的属性‘Interruptible’设置为‘yes’。
[*selection* , *ok*]= listdlg(‘ListString’, S,*prstr*,*alt*,...)	创建一个列表框。参数S是一个细胞矩阵，包含列表项。字符串**prstr**是窗口的属性，属性值设为*alt*。可以同时对多个属性进行设置，见helpdesk可得更多关于属性信息。如果给出向量**selection**，表示选中的列表项在列表S中的索引。如果给出 *ok*，用户按下 **OK**按钮，则*ok*=1；否则*ok*=0(在这种情况下**selection**是空矩阵)。
[*fname*, *path*]= uigetfile(filter, *title*, *x*, *y*)	创建一个打开文件的窗口。字符串 **filter**决定列出的文件；字符串可以包含“wildcards”或只是一个文件名，例如‘*m’列出所有M文件。字符串**title**是窗口的标题。参数*x*和*y*表示窗口在屏幕上的位置，有些系统可能不支持这个选项。如果给出 **fname**和**path**，它们分别保存的是所选文件的名称字符串和所选文件的路径名字符串。如果用户选中取消按钮或者有错误发生，就返回0。
[*fname*, *path*]= uiputfile(filter,*title*,*x*,*y*)	同上，但是是输入到文件中去。
pagedlg(*fig*)	创建一个对话框，用户可以在其中选择输出当前图形的纸张格式。可选参数*fig*表示纸张格式应用的图形*fig*，而不是当前图形。
printdlg(‘-crossplatform’, *fig*)	创建一个图形打印对话框，可选参数 *fig*表示要打印的图形数 *fig*，而不是缺省的当前图形。如果给出‘—crossplatform’，就使用一个MATLAB对话框，而不是操作系统的标准窗口。
uisetcolor(*x*, *title*)	创建一个窗口，用户可以从中选择颜色，可选参数 *x*可以是图形对象的句柄或者一个RGB三元组。如果没有指定*x*，就使用缺省颜色和黑色，字符串 **title**表示窗口的名字。命令返回一个RGB三元组。如果用户选择取消按钮或者发生错误，就返回0。
uisetfont(*pr*, *title*)	创建一个窗口，用户可以从中选择字体。字符串 **title**表示窗口的名字，可选参数 **pr**必须是一个轴对象或文本对象的句柄。选中的字体就应用在句柄为 **pr**的对象上。如果没有指定 **pr**，就返回一个应用选中字体的新文本对象句柄。如果用户选中取消按钮，就返回0。
btngroup(*h*,prstr, val, ...)	用来在句柄*h*的图形窗口中创建工具条，也就是按钮条。如果没有指定*h*，则使用当前窗口。可以通过设置工具条的属性来控制它的显示：将属性 **prstr**设置为*alt*，同时可以对多个属性进行设置。用help btngroup可



	得更多关于可用属性的信息。命令btndown、btnup、btnicon、btnpress和btnstate也可用来控制工具条，用help可得更多信息。
tabdlg	创建一个带有标记的对话框。使用help tabdlg可得更多信息。
waitbar(*x*, *title*)	创建一个进度显示条。参数*x*表示已完成的部分，字符串**title**表示窗口的名字。在显示工作进度时反复调用这个函数，同时*x*的值增长(从0到1)。返回窗口，即图形对象的句柄。当工作完成时可以用关闭按钮关闭这个窗口。

下列工具可以在图形对象中交互式地改变对象属性。

命令集169　交互式控制属性的工具

axlimdlg	一个图形用户界面，用来在轴对象中定义轴的坐标区间，使用help axlimdlg可得更多信息。
edtext	一个图形用户界面，用来在轴对象中交互式地编辑文本对象。使用help edtext可得更多信息。
plotedit	一个图形用户界面，用来添加文本和箭头到轴对象上，也可以用来改变轴对象的一些属性。使用help plotedit可得更多信息。

■ **例**14.16

创建一个有三个选项的菜单：

```
choice = menu('Choose:','Enter','Wait','Leave')
```

在XWindows环境中给出的窗口如图14-16所示。

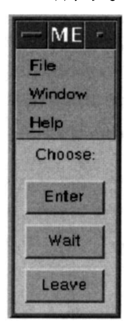


图14-16　用命令menu 创建菜单。因为这个菜单是细条形的，所以只能看见窗口标题‘MENU’的前两个字母

可以单击按钮‘Enter’、‘Wait’或‘Leave’进行选择。如果选中中间按钮，也就是‘Wait’，会返回得到：

```
choice =
          2
```

因为选中的是第二个按钮。

在不同的操作系统中，菜单看起来是不一样的。如果没有可用的图形，MATLAB就在命

令窗口中给出菜单选项并等待从键盘输入。

■

有时需要创建出高级图形用户界面，通常预定义窗口就不够用，必须定义新的窗口。这就要用到**Guide**(图形用户界面开发环境)工具包。这个工具包包含有五个程序用来创建窗口、菜单等。虽然可以使用set和get命令(见命令集154)，但是使用Guide工具包可以更有效地进行图形用户界面的开发。

命令集170 Guide工具包

ctpanel(*h*)	在Guide工具包中显示和其他程序相连接的控制板。在控制程序下给出可选句柄向量 **h**表示的图形对象，如果没有给出 **h**，则使用当前图形句柄。在这个程序中可以增加用户控制对象，如增加按钮和文本框到图形对象中。
guide(*h*)	同上。另外，Guide工具包的所有工具随着可选句柄向量 h表示的图形对象进行刷新。如果没有给出 **h**，则使用当前图形。
propedit(*h*)	打开属性编辑器，这个工具用来设置上节中列出的所有对象属性。参数h是对象句柄或句柄向量。如果没有给出 **h**，则使用当前对象句柄。如果句柄向量 **h**中的对象类型不同，则只列出它们的共有属性。
align	打开队列工具，这个工具可以用来排列用户控制对象和坐标系，以便它们都在直线上，并且之间有一定距离。这个工具可以分开使用；见helpdesk可得更多信息。
cbedit(*h*)	打开事件过程编辑器，这个工具可以用来决定当用户按下按钮，在文本框中输入文本时运行的回调函数。参数 **h**是一个对象句柄或者句柄向量。如果没有指定**h**，则使用当前对象句柄。
menuedit(*h*)	打开菜单编辑器。这个工具可以用来改变用户自定义菜单的一些属性。参数**h**是对象句柄或者句柄向量。如果没有指定 **h**，则使用当前对象句柄。

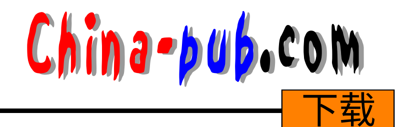
在设计一个图形用户界面时，开始要做的第一件事情是仔细考虑界面的外观和要完成的功能，最后很容易地使用Guide工具包中提供的工具来完成设计。

首先要完成程序的设计，用工具包来完成时就象面向对象的程序设计，它画出的窗口上可以带有按钮、弹出式菜单等。

完成界面绘制后的下一步是添加界面函数。当用户按下一个按钮时，希望会触发事件。界面函数可以由回调函数来实现，用 Guide工具包中的事件过程编辑器很容易地将它们放进图形对象中。

以上概括性地给出图形用户界面的开发过程。下面将对 Guide工具包中的每个工具 (见命令集170)进行介绍，以及用例程来进行说明。

在这个例程中，所有的屏幕画面都取自 X Windows环境，同样也可以用在 Windows和Macintosh系统中。实际上，在某个操作系统中创建的用户界面可以不做任何修改就能移植到另一个系统中。



在MATLAB手册《用MATLAB建立GUI》中可以找到更多关于建立图形用户界面的信息。这个手册还给出了在建立图形用户界面时要注意的一些问题。

14.3.1　控制面板

在MATLAB的命令窗口中键入命令`guide`，将启动控制面板，同时还打开一个被Guide工具包控制的图形窗口。网格表示该图形窗口当前的状态为受控状态，见图 14-18。如图形窗口处于打开状态，那么当前图形对象就由控制面板所控制。

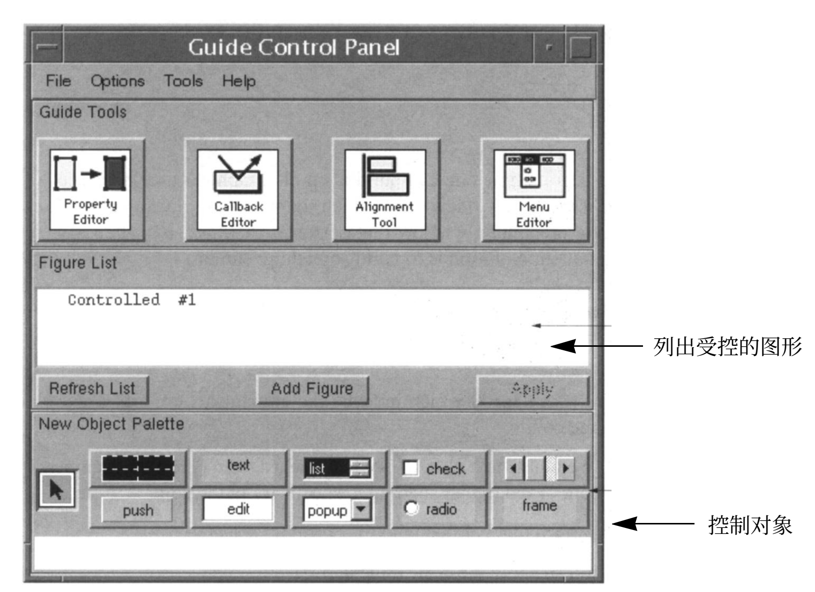


图14-17　控制面板

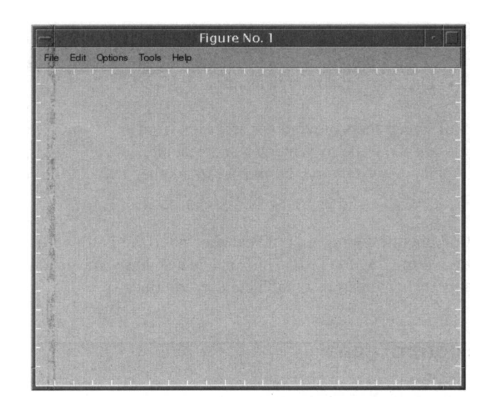


图14-18　Guide工具控制下的图形窗口

从图14-17中可以看出在Guide工具包中其他工具的连接关系，而且，有 10种不同类型的控制对象可以添加到图形窗口中来建立用户界面。

■ **例**14.17

这个例子说明了如何来建立一个简单用户界面，创建一个窗口，可以让用户在窗口中选择颜色。这实际上可以用预定义的窗口来完成，`uisetcolor`(见命令集168)可用于完成类似的任务。



首先决定要用的控制对象，下列对象可能都会用到：

- 三个滑标(‘slider’)，分别用来表示红、绿、蓝的颜色数。
- 三个可编辑的文本框(‘edit’)，用户可以分别给出所需要的三种颜色准确的颜色值。
- 一个帧(‘frame’)，用当前选中的颜色进行填充。
- 七个解释字符串(‘text’)。
- 两个按钮(‘push button’)：CANCEL和OK。

之后可以用控制面板来设计窗体。在控制面板中挑选对象后将它们拖到图形对象上。这一步只是给出窗体的大概设计；对象的大小和位置不必很准确。在图形窗口中可以进行对象的剪切、粘贴和复制。

图14-19中给出了控制对象在窗体的大概布置。

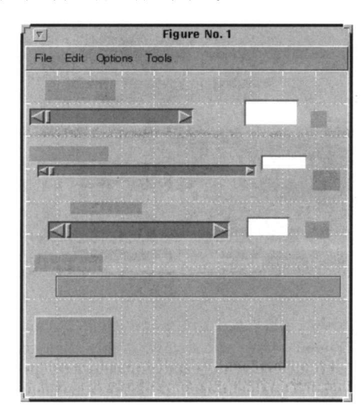


图14-19 控制对象的布置

使用下面两个工具来调整窗体。

■

14.3.2 属性编辑器

在控制面板中按下属性编辑器按钮，就会显示出下面的窗口来 (图14-20)：

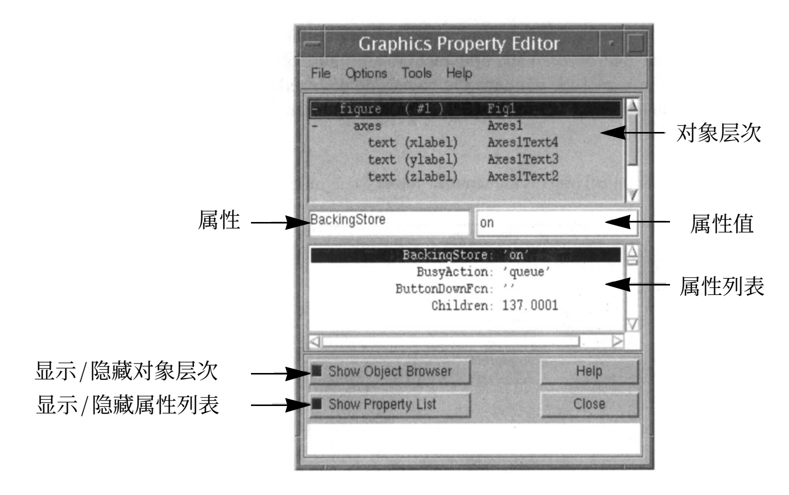


图14-20 属性编辑器，给出了对象的层次关系和属性列表

可以看出属性编辑器有四个部分，在窗口的顶部给出了对象层次关系 (见图14-4)，表示包含的所有对象以及对象之间的关系。其中左列字符表示对象是否有子对象：加号‘ + ’表示对象有子对象，但是未列出其子对象名字，如果点击对象名字时会显示出其子对象；减号‘ — ’表示对象有子对象，但是已经列出其子对象的名字。没有字符表示对象没有任何的子对象。选中一个或多个对象 (shift—左键)，在窗体的中间的属性列表中显示出它们的属性。可以在层次列表中选中对象，也可以在图形窗口中直接进行选择。

属性列表框中可以列出对象的所有属性。如果选中的对象类型不同，则只列出它们的共有属性。如果选中的对象中有些属性有不同的值，在列表框中以‘ Multiple Values’表示出来。选中一个属性，它的名字和相应的值就会在属性列表框上面的属性框和属性值框中分别给出。也可以不选择，而直接在属性框中输入属性名字。不必完全输入，因为有一些字符能唯一确定属性。例如，输入col，然后回车，属性编辑器就会给出属性Color和它的属性值。

显然，可以在属性框和属性值框中修改属性和属性值。如果属性只被赋予几个限定的值或文本区域，那么属性值框也可以是一个弹出式菜单。在文本区域中，属性编辑器忽略第一个空格之后的所有值，因此可以在行的开始处输入新值并按回车来修改属性值。

■ **例**14.18

接例14.17，利用属性编辑器可以对下列属性进行修改：

所有对象：

```
HandleVisibility:     'callback'
```

窗体(‘ **figure** ’)

```
MenuBar:              'none'
NumberTitle:          'off'
Name:                 'Color Picker'
Resize:               'off'
Tag:                  'ColPick'
```

滑标(‘ **slider** ’)

```
Position:             [?  ?  200 20]
Max:                  100
Value:                0
Tag:                  'RedSl', 'GreenSl', and 'BlueSl', respectively
```

滑标上的文本(‘ **text** ’)

```
Position:             [?  ?  50 16]
FontSize:             12
HorizontalAlignment:  left
BackgroundColor:      [0.8 0.8 0.8]
ForegroundColor:      'red', 'green', and 'blue', respectively
String:               'Red', 'Green', and 'Blue', respectively
```

滑标右边的可编辑文本框(‘ **edit** ’)

```
Position:             [?  ?  50 25]
String:               '0'
UserData:             0
FontSize:             12
```

```
HorizontalAlignment:  left
Tag:                  'RedTxt', 'GreenTxt', and 'BlueTxt', respectively
```

可编辑文本框右边的文本(‘text’)

```
Position:             [?  ?  16 16]
FontSize:             12
HorizontalAlignment:  left
BackgroundColor:      [0.8 0.8 0.8]
String:               '%'
```

颜色框(‘frame’)

```
Tag:                  'col'
BackgroundColor:      'black'
```

颜色框上的文本(‘text’)

```
Position:             [?  ?  50 16]
FontSize:             12
HorizontalAlignment:  left
BackgroundColor:      [0.8 0.8 0.8]
String:               'Color'
```

按钮(‘pushbutton’)

```
Position:             [?  ?  70 30]
FontSize:             14
FontWeight:           'bold'
String:               'CANCEL' and 'OK' respectively
Tag:                  'cancel' and 'OK' respectively
```

经过这些调整后，窗体如图14-21所示。

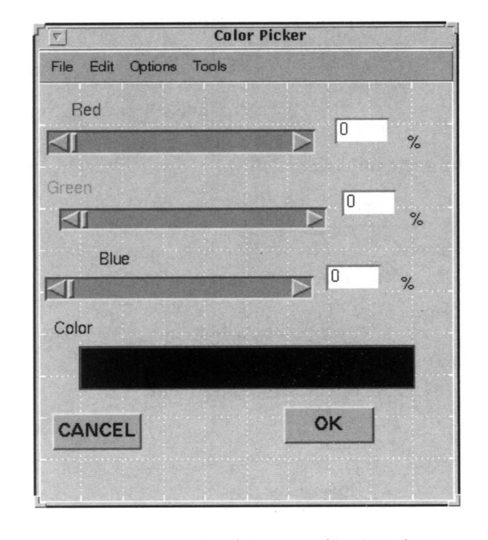
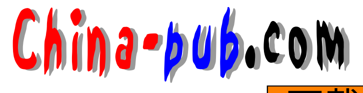


图14-21 用属性编辑器调整后的窗体

14.3.3 对齐工具

按下控制面板中的对齐工具按钮，会显示出如图14-22的窗体。



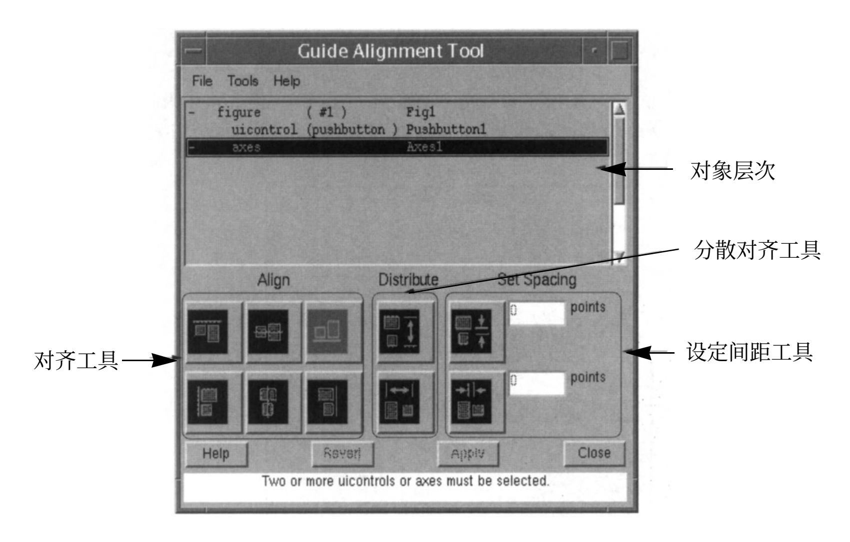


图14-22 对齐工具

在窗体的顶部列出了对象的层次列表，这和属性编辑器是一样的：可以在层次列表中选择对象，也可直接在图形窗口中选择。然而不同的是每次至少要同时选中两个对象。

当选中几个对象时就可以对它们进行相对调整，这可以通过选择调整工具来完成，也就是选择对象层次列表下的按钮。单击‘ Apply ’按钮进行确认，如果单击‘ Revert ’按钮，则取消调整设定。

有三组调整工具，第一组是对齐工具，它们将选中的对象在某个轴上对齐，六个工具都指定唯一轴。

第二组调整工具是分散对齐工具，它们将选中的对象在水平方向或垂直方向分散对齐，使得对象间的距离都是固定的。当在对象间调整时，最外面的两个对象保持原来位置不动。

第三组调整工具是设定指定的间距工具，它们可以在水平方向或垂直方向设定每对选中对象指定的间距。选中对象的调整从左下角开始，一直向上到右上角。注意，有些对象可能会超出图形窗口的范围。

■ **例**14.19

接例14.17，最后使用对齐工具来美化窗体。通过调整后，窗体如图 14-23所示。

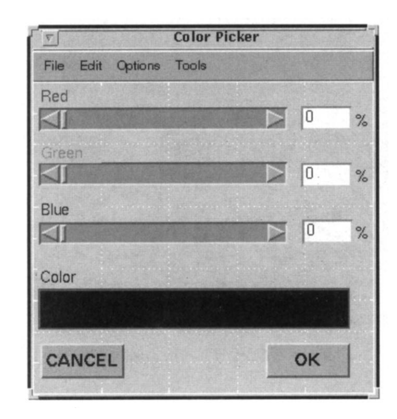


图14-23 经过对齐调整以后的图形

在控制面板中双击文本‘ Controlled #1:Color Picker ’可以激活这个界面。在界面被激活前，系统会给出提示是否保存界面，可以将它保存在文件 **ColorPicker.m**中，这是一个普通的M文件。在文本编辑器中可以查看文件内容。另外数据MAT文件被保存在文件**ColorPicker.mat**中。当文件保存后界面被激活，此时可以拖动滑标，点击按钮等，但是没有任何事件发生。这是因为没有编写界面函数，所以要用到下一个工具：事件过程编辑器。

■

14.3.4 事件过程编辑器

在控制面板中单击“ Callback Editor ”按钮，打开如图14-24所示的事件过程编辑器。

事件过程编辑器的顶部是对象层次列表，和属性编辑器一样。可以在层次列表中或直接在图形中选择要编写回调函数的对象。

对象有对应着不同事件的不同类型回调函数。例如创建或撤消一个对象、用户用鼠标点击对象，这些都是事件。通过回调函数菜单来选择回调函数类型，见图 14-24。

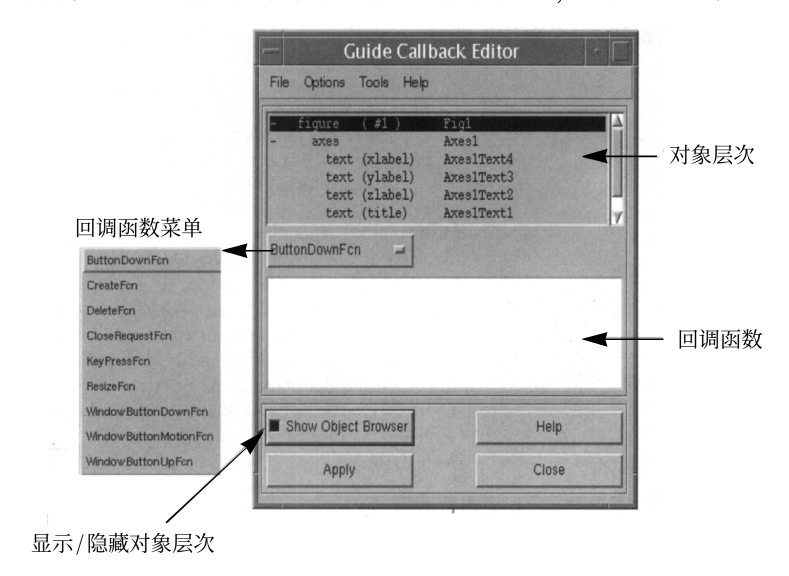


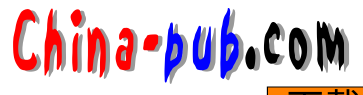
图14-24 事件过程编辑器

当选定某个类型的回调函数后，在事件过程编辑器的中间位置文本区域内书写函数体，和在MATLAB命令窗口中一样。除非函数特别的短，比较方便的做法是调用一个包含事件过程函数代码的M文件。值得注意的是如果在 M文件中做了改动，比如调试时，那么这些改动只有在MATLAB命令窗口中输入命令`clear functions`后才有效。

在编写事件过程函数时有许多很有用的MATLAB命令。

命令集171 事件过程命令

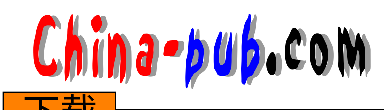
`[h,fig]=gcbo`	返回当前正在执行回调函数的对象句柄 *h*。如果没有函数在执行，返回空矩阵[]。可选参数*fig*中是返回的父图形对象句柄。
`gcbf`	返回当前正在执行回调函数的对象的父对象句柄。如果没有函数在执行，返回空矩阵 []。如果回调函数属于图形对象，



	则返回这个对象的句柄。
`selectmoversize`	使用时和`ButtonDownFcn`相同。当执行时对图形对象有下列动作：选择、移动、复制和重定义大小。返回一个结构 A，其中域**A.Type**可以是‘`Select`’、‘`Move`’、‘`Resize`’或‘`Copy`’，域**A.Handles**可以是被选中对象(‘`Select`’)的句柄向量，也可以是一个 *m*×2矩阵。矩阵中的第一列是原始对象的句柄，第二列是复制后新对象的句柄(‘`Copy`’)。
`overobj(type)`	在鼠标指针下搜索可视的图形对象，字符串 **type**给出搜索图形对象的类型。这个命令使用的前提是根对象的属性 `Units` 设置为‘`pixels`’，而且只适用于有子对象的图形对象上。
`setstatus(h,str)`	设置用户控制对象的属性 `String`，其中用户对象的属性`Style`设置为‘`text`’，属性`Tag`设置为‘`Status`’；见表14-15。用户控制对象必须是句柄*h*图形对象的子对象。
`getstatus(h)`	和上个命令相似，但是返回属性`String`。
`hidegui(`*`h,status`*`)`	设置句柄*h*图形对象的属性`HandleVisibility`；见表14-2。如果没有指定*h*，则使用当前对象句柄。参数**status**是一个可选字符串，可以为缺省值‘`on`’、‘`off`’或‘`callback`’。这个命令主要用来保护用户界面，防止用户通过命令窗口来撤消它。
`status=hidegui(`*`h`*`)`	返回得到句柄*h*图形对象的属性`HandleVisibility`。如果没有指定*h*，则使用当前对象句柄。
`waitfor(h,`*`prstr,alt`*`)`	阻碍函数调用直到撤消句柄*h*图形对象或用户按下`Control C`。同时，如果给出可选属性 **prstr**，这个属性将被改变。而且，如果给出*alt*，就将**prstr**的值设为*alt*。如果*h*不是一个句柄，**prstr**不是一个合法的属性或*alt*不是一个有效的值，则这个命令不起作用。
`uiwait(`*`h`*`)`	阻碍函数调用，事件过程函数必须存在于句柄 *h*图形对象的一个控制对象中，也就是撤消图形对象或调用 `uiresume`。如果没有指定*h*，就用当前图形对象句柄。
`uiresume(h)`	继续执行被`uiwait`阻碍的函数，见上。
`status=uisuspend(h)`	挂起所有与句柄*h*图形对象中控制对象的用户交互。例如，单击按钮不再有事件过程函数被调用。返回得到的结构**status**中包含图形对象的属性信息。用`uirestore`可以重复取消挂起。
`uirestore(status)`	取消被 `uisuspend`挂起的用户交互，参数 **status**是`uisuspend`返回得到的结构。

■ **例**14.20

接例14.17。通过双击控制面板中的文本‘ Active #1:Color Picker ’来设置Guide工具控制



下的图形窗口。在事件过程编辑器中编写下列‘ Callback ’类型的事件过程函数：

红色滑标：　　CbColPic redsl;
红色文本区域：　　CbColPic redtxt;
绿色滑标：　　CbColPic greensl;
绿色文本区域：　　CbColPic greentxt;
蓝色滑标：　　CbColPic bluesl;
蓝色文本区域：　　CbColPic bluetxt;
Cancel按钮：　　CbColPic cancel;
Ok按钮：　　CbColPic ok;

下面编写M文件CbColPic.m：

```
function CbColPic(action)

switch(action)

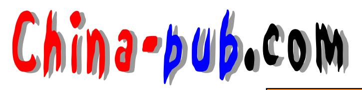
  case 'redsl'                  % 红色滑标
                                % 取红色文本框的指针
    RedTxtPtr = findobj(gcbf,'Tag','RedTxt');
                                % 读取滑标值
    Val = get(gcbo,'Value');
                                % 设置红色文本框为滑标值
    set(RedTxtPtr,'String',num2str(Val,3));
                                % 刷新框的颜色
    LocalUpdateCol;

  case 'greensl'                % 绿色滑标

  GreenTxtPtr = findobj(gcbf,'Tag','GreenTxt');
  Val = get(gcbo,'Value');
  set(GreenTxtPtr,'String',num2str(Val,3));
  LocalUpdateCol;

case 'bluesl'                 % 蓝色滑标
  BlueTxtPtr = findobj(gcbf,'Tag','BlueTxt');
  Val = get(gcbo,'Value');
  set(BlueTxtPtr,'String',num2str(Val,3));
  LocalUpdateCol;

case 'redtxt'                 % 红色文本框
                              % 取红色滑标指针
  RedSlPtr = findobj(gcbf,'Tag','RedSl');
                              % 取文本框中字符串
  Str = get(gcbo,'String');
                              % 转换字符串
  Val = LocalChStr(Str);
                              % 设置滑标值
  set(RedSlPtr,'Value',Val);
                              % 设置文本框中字符串格式
  set(gcbo,'String',num2str(Val,3));
```




```
                                           % 将值保存到UserData中
    set(gcbo,'UserData',Val);
                                           % 刷新颜色框
    LocalUpdateCol;

  case 'greentxt'                          % 绿色文本框
    GreenSlPtr = findobj(gcbf,'Tag','GreenSl');
    Str = get(gcbo,'String');
    Val = LocalChStr(Str);
    set(GreenSlPtr,'Value',Val);
    set(gcbo,'String',num2str(Val,3));
    set(gcbo,'UserData',Val);
    LocalUpdateCol;

  case 'bluetxt'                           % 蓝色文本框
    BlueSlPtr = findobj(gcbf,'Tag','BlueSl');
    Str = get(gcbo,'String');
    Val = LocalChStr(Str);
    set(BlueSlPtr,'Value',Val);
    set(gcbo,'String',num2str(Val,3));
    set(gcbo,'UserData',Val);
    LocalUpdateCol;

end
function OutVal = LocalChStr(InStr)       % 转换字符串

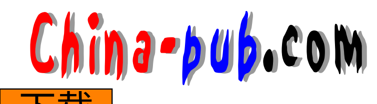
temp = str2num(InStr);                     % 将字符串转换为数字

if (isempty(temp))                         % 字符串不是一个数字串
  OutVal = get(gcbo,'UserData');           % 使用前次值
elseif (temp > 100)                        % 数字大于100
  OutVal = 100;
elseif (temp < 0)                          % 数字小于0
  OutVal = 0;
else
  OutVal = temp;
end

function Col = LocalUpdateCol              % 刷新颜色框

% 滑标的指针
RedPtr   = findobj(gcbf,'Tag','RedSl');
GreenPtr = findobj(gcbf,'Tag','GreenSl');
BluePtr  = findobj(gcbf,'Tag','BlueSl');

% 取滑标值
RedVal   = get(RedPtr,  'Value')/100;
GreenVal = get(GreenPtr,'Value')/100;
BlueVal  = get(BluePtr, 'Value')/100;
Col = [RedVal,GreenVal,BlueVal];
```




```
% 取颜色框指针
FrPtr = findobj(gcbf,'Tag','col');
% 设置它的颜色
set(FrPtr,'BackgroundColor',Col);
```

在这个窗体中，要让滑标和它相应的文本区表示出相同的颜色数，比如红色，所以必须将它们两者关联起来。这样，当颜色滑标移动时，相应的文本区也得到刷新。

这样必须写两个函数来完成这样的工作：

• `LocalChStr(InStr)` 将字符串InStr转换成数字。这个函数用来转换用户在文本区内输入的数字字符串。

• `LocalUpdataCol` 设置颜色框的颜色。

通过双击控制面板上的文本‘ Controlled #1:Color Picker ’来激活用户界面和进行保存。颜色的设定可以通过拖动滑标或者在文本区内直接输入数值，颜色框上就可以将选中的颜色显示出来！见图14-25。

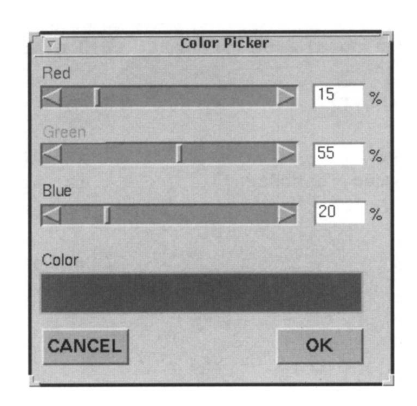


图14-25 通过编写事件过程函数来激活窗体

现在还没有编写鼠标事件，在下一小节中将会介绍。

■

14.3.5 菜单编辑器

在控制面板上按下‘ Menu Editor ’按钮，就会打开如图14-26所示的菜单编辑器。

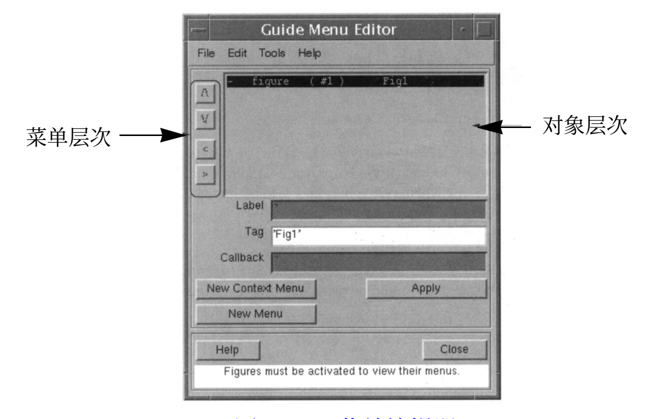
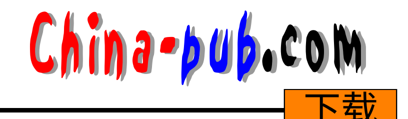


图14-26 菜单编辑器



在菜单编辑器的顶部列出了对象的层次结构关系，它和其他 Guide工具一样。所不同的是只显示出图形和菜单对象。

菜单编辑器的其他部分及其功能：

- ‘New Menu’按钮用来添加图形对象菜单或已有菜单的菜单选项。
- ‘New Context Menu’按钮用来创建对象的快捷菜单；见14.2.8节。
- ‘Label’文本区用来输入菜单或菜单选项标签，这个字符串将会在用户界面中显示。
- ‘Tag’文本区用来输入分配给菜单的标记符，在编写事件过程函数时要用到这个标记符。
- ‘Callback’文本区用来输入菜单的事件过程函数，也就是在用户选择菜单时要运行的命令或M文件。事件过程函数必须是字符串，和属性编辑器不一样 (见前小节)。
- 对象层次结构关系左边的四个按钮是用来在图形窗口中对菜单标记的，可以移动菜单或菜单中的选项。

■ **例**14.21

接例14.17。借助菜单编辑器来定义按钮‘CANCEL’和‘OK’的事件过程函数，但是首先要添加一个可用的菜单：预定义颜色菜单。

通常在控制面板中双击文本‘Active #1:Color Picker’来调出窗体。创建一个新菜单，分配给它下列数据：

```
Label: 'Color'             Tag: ''   Callback: ''
Label: 'Pink'              Tag: ''   Callback: 'CbColPic menpink;'
Label: 'Military Green'    Tag: ''   Callback: 'CbColPic menmilgreen;'
Label: 'Beige'             Tag: ''   Callback: 'CbColPic menbeg;'
```

在M文件CbColPic.m中添加菜单选项的事件过程函数：

```
function CbColPic(action)

switch(action)

...
  case 'menpink';                  % 粉红色
    LocalUpdateAll([1 0.4 0.7]);

  case 'menmilgreen'               % 绿色
    LocalUpdateAll([0.15 0.55 0.2]);

  case 'menbeg'                    % 浅褐色
    LocalUpdateAll([1 0.9 0.8]);

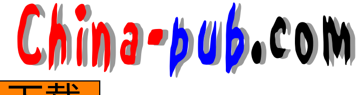
end

...

function LocalUpdateAll(Color)     % 全部刷新

Color = 100.*Color;

% 获取滑标指针并刷新
```




```
RedSlPtr    = findobj(gcbf,'Tag','RedSl');
GreenSlPtr = findobj(gcbf,'Tag','GreenSl');
BlueSlPtr  = findobj(gcbf,'Tag','BlueSl');
set(RedSlPtr,'Value',Color(1));
set(GreenSlPtr,'Value',Color(2));
set(BlueSlPtr,'Value',Color(3));

% 获取文本框指针并设置

RedTxtPtr   = findobj(gcbf,'Tag','RedTxt');
GreenTxtPtr = findobj(gcbf,'Tag','GreenTxt');
BlueTxtPtr  = findobj(gcbf,'Tag','BlueTxt');
set(RedTxtPtr,'String',num2str(Color(1),3));
set(GreenTxtPtr,'String',num2str(Color(2),3));
set(BlueTxtPtr,'String',num2str(Color(3),3));

% 获取颜色框指针并设置颜色

FrPtr = findobj(gcbf,'Tag','col');
set(FrPtr,'BackgroundColor',Color./100);
```

注意，现在已经编写了刷新滑标、文本区和颜色框的函数。

通过双击控制面板上的文本‘ Controlled #1:Color Picker ’激活用户界面并进行保存。现在也可以用菜单来选择颜色了，见图14-27。

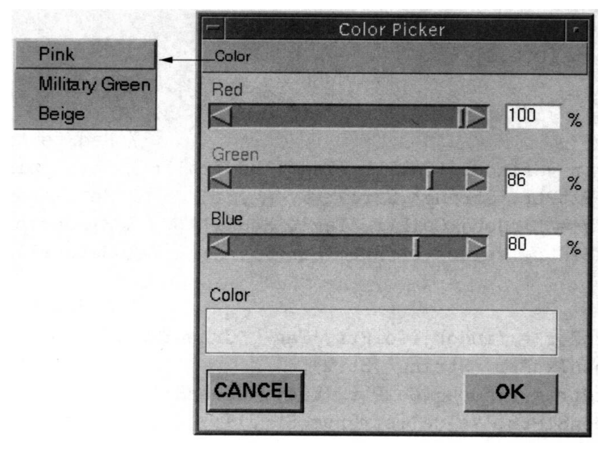
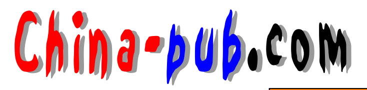


图14-27　添加菜单后的窗体

最后来看一下‘ CANCEL ’和‘ OK ’按钮。当用户按下‘ OK ’按钮时可以返回选定的颜色，所以还得在程序中编写相应的程序代码。而且，希望在开始时有一个初始颜色。如果用户按下‘ CANCEL ’按钮，就返回初始颜色。这样就要编写一个新函数，保存在 M文件**uicolor.m**中：

```
function rgbout = uicolor(rgbin)

ColorPicker;                              % 打开颜色混合器
set(0,'ShowHiddenHandles','on');          % 获取指针
ColPtr = findobj(0,'Tag','ColPick');
```




```
ColPtr = ColPtr(1);                                 % 最新创建

if (nargin >= 1)                                    % 检查输入内容
  if ((length(rgbin) ~= 3) | (min(rgbin) < 0) | ...
               (max(rgbin) > 1))                    % 警告：非法输入
    warndlg('Invalid Colorspecific.','Warning');
    Str1 = '0'; Str2 = '0'; Str3 = '0';             % 用标准值代替
    InCol = [0,0,0];
  else                                              % 输入
    Str1 = num2str(rgbin(1) * 100,3);
    Str2 = num2str(rgbin(2) * 100,3);
    Str3 = num2str(rgbin(3) * 100,3);
    InCol = rgbin;
  end
else
                                                    % 如果没有输入，就用标准值

  Str1 = '0'; Str2 = '0'; Str3 = '0';
  InCol = [0,0,0];
end

                                                    % 红色文本框
RedTxtPtr = findobj(ColPtr,'Tag','RedTxt');         % 获取指针
set(RedTxtPtr,'String',Str1);                       % 设置框体
RedSlPtr = findobj(ColPtr,'Tag','RedSl');           % 滑标指针
set(RedSlPtr,'Value',str2num(Str1));                % 设置滑标

                                                    % 绿色文本框
GreenTxtPtr = findobj(ColPtr,'Tag','GreenTxt');
set(GreenTxtPtr,'String',Str2);
GreenSlPtr = findobj(ColPtr,'Tag','GreenSl');
set(GreenSlPtr,'Value',str2num(Str2));

                                                    % 蓝色文本框
BlueTxtPtr = findobj(ColPtr,'Tag','BlueTxt');
set(BlueTxtPtr,'String',Str3)
BlueSlPtr = findobj(ColPtr,'Tag','BlueSl');
set(BlueSlPtr,'Value',str2num(Str3));

FrPtr = findobj(ColPtr,'Tag','col');                % 颜色框指针
set(FrPtr,'BackgroundColor',InCol);                 % 设置颜色框

% 将输入保存在UserData中
set(ColPtr,'UserData',InCol);

uiwait(ColPtr);                                     % 等待响应
rgbout = get(ColPtr,'UserData');                    % 获取选中的颜色
close(ColPtr);                                      % 关闭颜色混合器
set(0,'ShowHiddenHandles','on');                    % 隐藏指针
```

上面的程序可以停下来等用户选择颜色。初始颜色，一个 RGB三元组，保存在颜色混合

器的UserData中，用户进行颜色选择时也从这里来选取。所以只需调用‘ OK ’按钮的事件过程函数来覆盖这个颜色就可以，因为如果用户按下‘ CANCEL ’按钮，返回的是初始颜色。在M文件CbColPin.m中添加两个事件过程函数：

```
function CbColPic(action)

switch(action)

...
  case 'cancel'
    uiresume(gcbf);

  case 'ok'
    Col = LocalUpdateCol;
    set(gcbf,'UserData',Col)
    uiresume(gcbf);

end

...
```

现在整个程序就全部完成了，可以调用**uicolor**来使用它(不是**ColorPicker**，因为用它只定义了窗体本身)。下面的这个例子用来说明如何使用这个程序来改变坐标轴中图形窗口的前景色和背景色。

```
function change

fp = figure;                      % 创建一个图形窗口
ap = axes;                        % 创建坐标系

InCol = get(fp,'Color');
OutCol = uicolor(InCol);          % 设置窗口的背景色
set(fp,'Color',OutCol);

InCol = get(ap,'Color');          % 设置坐标系的前景色
OutCol = uicolor(InCol);
```

■

14.4 动画

14.4.1 介绍性示例

本节开始先介绍一个例子，做一个简短动画来演示例 13.8中的曲线振荡效果。

■ **例**14.22

```
x        = -8:0.5:8;              % 定义曲面
[XX,YY] = meshgrid(x);

r = sqrt(XX.^2+YY.^2) + eps;
Z = sin(r)./r;
```




```
surf(Z);                          % 画出帧

% 保存坐标值，使得所有帧都在同一个坐标系中
% 创建一个保存动画的矩阵，保存20帧

theAxes = axis;
fmat = moviein(20);

% 循环创建动画数据

for j = 1:20
  surf(sin(2*pi*j/20)*Z,Z)        % 画出每一步的曲面
  axis(theAxes)                   % 使用相同的坐标
  fmat(:,j) = getframe;           % 拷贝帧到矩阵中
end

movie(fmat,10);                   % 演示动画10次
                                  % 这很有趣!
```

可以看出，动画可以以帧的形式保存在矩阵的每一列中，之后再从矩阵中显示出来。在记录动画前要保存坐标轴的最小和最大值，这样就可以使每一帧都有相同的坐标轴。

■

14.4.2 拷贝图形窗口

命令getframe用来拷贝一个图形窗口到一个列向量中。

命令集172 拷贝图形

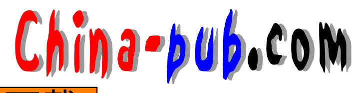
getframe(*p*, *r*)	拷贝句柄*p*对象中的图像。对象可以是屏幕、图形或坐标系。如果没有指定*p*，则使用当前图形窗口。可选参数r表示要拷贝对象的矩形区域。r是一个位置向量，有四个元素[*left bottom width height*]。
[X,Ftab]=getframe(···)	返回图像矩阵**X**和色表**Ftab**。
[X,Ftab]=frame2im(F)	将动画帧**F**转换成图像矩阵**X**，**X**的元素在色表**Ftab**中给出。

14.4.3 创建动画

在记录动画前要给它的帧分配空间，这些帧以列的形式存储在矩阵中，用命令 moviein 可以创建这样的矩阵。

命令集173 制作动画

moviein(n)	创建一个可以保存*n*帧的矩阵。帧的大小由当前图形窗口的大小来决定，结果是必须先画出其中的一帧；见例 14.22。



`F= im2frame(X,Ftab)`	将图像矩阵**X**转换成一帧**F**，**X**矩阵的元素在色表**Ftab**中给出。这个命令可以用来从一个图形序列中创建动画。

14.4.4 演示动画

动画的演示借助`movie`命令，这个命令可以带上参数来表示动画演示的方式和位置。

命令集174 演示动画

`movie(`*`p`*`,Mat,`*`n`*`,`*`fps`*`,`*`pos`*`)`	连续演示*n*次动画矩阵**Mat**，每秒*fps*帧。如果**n**是一个整数向量，帧就按向量中的顺序来显示。可选参数 *p*是图形或轴对象的句柄，表示动画演示的位置。如果指定可选参数**pos**=(*x*, *y*)，动画就在这个位置显示，*x*和*y*是`Units`中的单位，相对于对象*p*的左下角的位置。



第15章　MATLAB与其他编程语言结合

MATLAB可以和其他编程语言一起使用，可以调用 FORTRAN或C程序。反过来FORTRAN或C也可以调用MATLAB程序。这样，快速的编译程序就可以利用MATLAB中强大的矩阵或图形命令，通过编写部分的C或FORTRAN程序，并进行编译，就可以避免MALTAB程序的瓶颈现象。

MATLAB还可以结合使用其他的应用程序，如Microsoft Word for Windows。这将在本章的最后一节讨论，这主要取决于计算机的系统和安装的应用程序。

15.1　介绍MATLAB和FORTRAN或C

MATLAB可以被FORTRAN或C语言程序调用，它也可以调用 FORTRAN或C语言程序。如果MATLAB程序运行速度很慢，后者对此很有用。因为 MATLAB是一个解释性语言，所以当运行程序时就是解释它的命令。这样有时会导致程序的运行速度很慢，如 for-loops循环。

在FORTRAN 77和C中可以使用MATLAB库，也可以用FORTRAN 90或C++对它们进行链接。

除非特别需要，一般不推荐编写 FORTRAN或C程序。MATLAB的优点在于可以用高级的形式描述出操作，而程序员不必担心循环的次数和一些其他细节问题。

被MATLAB调用的程序必须在编译后转换成 MEX文件，这样才能被MATLAB调用。在编译时它们和M文件一样使用。

在2.8节中讲到了由MATLAB创建的二进制文件。它们是以 MAT文件形式被调用的，在 C或FORTRAN语言的库中有用来读和写二进制文件的程序。注意，这些文件可以在不同的平台间传递，例如，用户可以读取在 Windows环境下建立的MAT文件到UNIX环境中。在15.4节中介绍了如何在MATLAB中读或写其他的二进制文件。这对有特殊格式要求的程序很有用。

MATLAB编译器、C数学库和C++数学库可以从MathWorks公司买到。首先可以作为自动MEX文件生成器或C源代码生成器使用，结合C数学库一起生成应用程序。

在C中编写MATLAB程序，数据通过指针来访问。在其他编程语言中调用 MALTAB程序，就要求使用指针。

在MATLAB 5中，所有变量类型，如标量、向量、矩阵、字符串、细胞矩阵和结构，都以mxArrays形式来保存，所有的数据操作都通过这些 mxArrays来完成。

MATLAB 5中新的数据类型，也就是多维数组、细胞矩阵和结构只能在 C中使用，而不能在FORTRAN中使用。

在C或FORTRAN中使用的MATLAB程序主要分四类：

`mx`	可操作的mxArrays。
`mat`	MAT文件。
`eng`	MATLAB工程文件。
`mex`	MEX程序，在MALTAB环境中完成一些操作的程序。

在下面几节中将举一些例子。这些例子基本上说明了 MATLAB和C或FORTRAN是如何相

互调用的。它们已在运行Sun OS 5.5.1的工作站上和Soloais CDE 1.0.2版的Windows系统中编译通过。对于每一种系统而言，编程的思想都是一样的。然而还是有一些重要的细节方面是不相同的。这就是为什么MATLAB中的例程很有趣的原因，它们可以在库中找到：

```
.../matlab52/extern/examples
```

路径中的三个点，...，表示这部分路径与系统有关。

例程的文档可以用MATLAB命令`helpdesk`获得。而且还有MATLAB手册《应用程序接口指南》。

15.2　MATLAB和C

为了使C和MATLAB混合编程，重要的是使用的C编译器以ANSI C标准进行编译。

15.2.1　C中对mxArray的操作

用下面描述的程序可以对mxArray进行操作。为了使用这些程序，在程序中必须嵌入头文件matrix.h，也就是在程序的开始包含下面一行：

```
#include "matrix.h"
```

下面表中的程序用来分配和释放内存。一个好的编程习惯就是及时释放不再使用的内存。不必使用MATLAB程序来创建数据结构，因为在程序结束时MATLAB会自动地来完成(可见命令集195中程序`mexMakeArrayPersistent`和`mexMakeMemoryPersistent`)。

命令集175　C中的内存管理

```
void *mxCalloc(size_t n, size_t size);
```

分配内存。参数*n*表示分配的元素个数，*size*表示每个元素的字节数。如果分配成功，返回一个指向已分配内存的开始位置的指针；否则返回 NULL。在程序中必须嵌入库文件`<stdlib.h>`。当不再使用时用`mxFree`来释放内存。

```
void mxSetAllocFcns(calloc_proc callocfcn,free_proc
freefcn,realloc_proc reallocfcn, malloc_proc mallocfcn);
```

在非MEX程序中用来释放内存。使用`helpdesk`可得更多信息。

```
void mxFree(void *ptr);
```

释放*ptr*指向的内存空间。

```
void *mxRealloc(void *ptr, size_t size);
```

重新分配用`mxCalloc`分配的内存。参数*ptr*是指向内存开始位置的指针，*size*是分配元素的个数。如果分配成功，返回得到指向分配内存开始位置的指针；否则返回NULL。在程序中必须嵌入库文件`<stdlib.h>`。用`mxFree`来释放不再使用的内存。

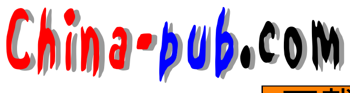
```
void mxDestroyArray(mxArray *array_ptr);
```

释放*array_ptr*指向的mxArray内存。

下面的常用程序用来管理和检查mxArray，如命名、重构和检查它们的类型。

命令集176　C中处理mxArray的常用程序

```
mxComplexity
```



是一个枚举数据类型，用来表示 mxArray的虚数元素。它的值可以为 mxCOMPLEX(复数mxArray)或mxREAL(其他)。

mxClassID

是一个枚举数据类型，用来表示 mxArray的类型。有下列选项：

```
mxCELL_CLASS,          细胞类型。
mxSTRUCT_CLASS,        结构类型。
mxOBJECT_CLASS,        用户自定义类型。
mxCHAR_CLASS,          字符串类型。
mxSPARSE_CLASS,        稀疏矩阵。
mxDOUBLE_CLASS,        双精度浮点小数。
mxSINGLE_CLASS,        单精度浮点小数。
mxINT8_CLASS,          8位整数。
mxUINT8_CLASS,         8位无符号整数。
mxINT16_CLASS,         16位整数。
mxUINT16_CLASS,        16位无符号整数。
mxINT32_CLASS,         32位整数。
mxUINT32_CLASS,        32位无符号整数。
mxUNKNOWN_CLASS,       未知类型。
```

`mxClassID mxGetClassID(const mxArray *array_ptr);`

返回*array_ptr*指向的mxArray类型；见上。

`const char *mxGetClassName(const mxArray *array_ptr);`

同上，返回字符串形式的类型。

`bool mxIsClass(const mxArray *array_ptr, const char *name);`

如果*array_ptr*指向的mxArray有字符串**name**表示的类型，则返回真。字符串 name相对应于上面的类型(见mxClassID)：“cell”、“struct”、“char”、“sparse”、“double”、“single”、“int8”、“uint8”、“int16”、“uint16”、“int32”和“uint32”。它还可以是自定义的类型名。

`const char *mxGetName(const mxArray *array_ptr);`

返回包含*array_ptr*指向的mxArray名字的字符串。

`double mxGetScalar(const mxArray *array_ptr);`

返回*array_ptr*指向的mxArray的第一个实数元素的值。总是返回一个 double型值。如果mxArray是一个结构或细胞类型，则返回 0.0；如果mxArray是一个稀疏矩阵类型，则返回第一个非零实数元素的值；如果 mxArray为空，则返回一个不确定值。

`mxArray *mxDuplicateArray(const mxArray *in);`

复制*in*指向的 mxArray，并返回指向复制 mxArray的指针。当它不再使用时，用mxDestroyArray来释放它；见命令集175。

`int mxGetNumberOfElements(const mxArray *array_ptr);`

返回*array_ptr*指向的mxArray的元素个数。使用mxGetClassID来找出元素类型。

`int mxGetElementSize(const mxArray *array_ptr);`

返回保存*array_ptr*指向的mxArray中一个元素需要的字节数。如果 mxArray是细胞或结构类型，则返回指向它们的指针大小。如果操作失败，返回 0。

```
int mxGetNumberOfDimensions(const mxArray *array_ptr);
```

返回*array_ptr*指向的mxArray中的维数，这个数总是不小于2。

```
const int *mxGetDimensions(const mxArray *array_ptr);
```

返回一个整数向量的指针，包含 *array_ptr*指向的mxArray的每一维的元素个数。

```
int mxSetDimensions(mxArray *array_ptr, const int *size, int ndims);
```

用来重构或增加 /减少 *array_ptr*指向的 mxArray的元素。参数 *ndims*表示维数范围，*size*表示一个整数向量的指针，包含每维中需要的元素个数。如果操作成功，返回0；否则返回1。如果要增加或减少元素，则必须进行分配 /释放内存。用helpdesk可得更多信息。

```
int mxGetM(const mxArray *array_ptr);
```

返回‘行’数，也就是*array_ptr*指向的mxArray的第一维中元素的个数。

```
void mxSetM(mxArray *array_ptr, int m);
```

用来重构或增加/减少*array_ptr*指向的mxArray中的‘行’数。参数m表示规定的‘行’数，见mxSetDimensions。

```
int mxGetN(const mxArray *array_ptr);
```

返回‘列’数，也就是*array_ptr*指向的mxArray的第二维中元素的个数。

```
void mxSetN(mxArray *array_ptr,int n);
```

用来重构或增加/减少*array_ptr*指向的mxArray中的‘列’数。参数*n*表示规定的‘列’数，见mxSetDimensions。

```
bool mxIsEmpty(const mxArray *array_ptr);
```

如果*array_ptr*指向的mxArray为空，就返回真。

```
bool mxIsFromGlobalWS(const mxArray *array_ptr);
```

如果*array_ptr*指向的mxArray是从MATLAB全局工作区中复制得到，则返回真。

```
bool mxIsNumeric(const mxArray *array_ptr);
```

如果*array_ptr*指向的mxArray是数字或字符串类型，则返回真。

```
bool mxIsInt8(const mxArray *array_ptr);
```

8位整数。

```
bool mxIsUint8(const mxArray *array_ptr);
```

8位无符号整数。

```
bool mxIsInt16(const mxArray *array_ptr);
```

16位整数。

```
bool mxIsUint16(const mxArray *array_ptr);
```

16位无符号整数。

```
bool mxIsInt32(const mxArray *array_ptr);
```

32位整数。

```
bool mxIsUint32(const mxArray *array_ptr);
```

32位无符号整数。

```
bool mxIsSingle(const mxArray *array_ptr);
```



单精度浮点小数。

```
bool mxIsDouble(const mxArray *array_ptr);
```

双精度浮点小数。

```
bool mxIsComplex(const mxArray *array_ptr);
```

复数。如果*array_ptr*指向的mxArray按函数指定的格式存储数据，则返回真。

```
int mxCalcSingleSubscript((const mxArray *array_ptr,int nsbus,int *subs)
```

将多维中的坐标向量转换成字典序中的标量下标。参数 *nsubs*通常表示*array_ptr*指向的mxArray中的维数，*subs*表示要转换坐标向量的指针。用helpdesk可得更多信息。

下面的程序用来创建和处理二维 $m \times n$满矩阵，矩阵的元素是双精度浮点小数。

命令集177 C中满矩阵的处理

```
mxArray *mxCreateDoubleMatrix(int m, int n, mxComplexity Complexflag);
```

和mxCreateCellMatrix相似(见命令集181)，但是这里创建的是二维 $m \times n$双精度浮点小数矩阵。如果矩阵中元素有复数，则参数 *Complexflag*是mxCOMPLEX类型，否则是mxREAL类型。

```
double *mxGetPr(const mxArray *array_ptr);
```

返回*array_ptr*指向的mxArray中第一个实数元素的指针。如果矩阵中没有任何实数元素，则返回NULL。

```
void mxSetPr(mxArray *array_ptr, double *pr);
```

设置*array_ptr*指向的mxArray中的实数元素。参数*pr*包含应该使用值的向量指针，这个向量必须用mxCalloc来动态地分配；见命令集 175。

```
double *mxGetPi(const mxArray *array_ptr);
```

和mxGetPr相似，但是是对虚数元素。

```
void mxSetPi(mxArray *array_ptr, double *pi);
```

和mxSetPr相似，但是是对虚数元素。

下面的程序用来创建和处理二维 $m \times n$的稀疏矩阵，矩阵元素是双精度浮点小数。

命令集178 C中稀疏矩阵的处理

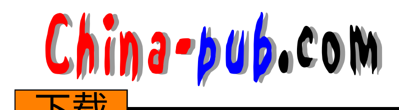
```
mxArray *mxCreateSpares(int m, int n int nzmax, mxComplexity ComplexFlag);
```

创建一个二维$m \times n$的稀疏矩阵。参数*nzmax*表示矩阵中非零元素的个数。如果矩阵中有复数元素，则参数*ComplexFlag*是mxCOMPLEX类型；否则是mxREAL类型。如果创建成功，返回指向这个矩阵的指针；否则返回 NULL。当它不再使用时，用mxDestroyArray来释放所占内存；见命令集175。

```
int mxGetNzmax(const mxArray *array_ptr);
```

返回*array_ptr*指向的稀疏矩阵 mxArray中的*nzmax*值(见上)。如果发生任何错误，都返回一个不确定数。

```
void mxSetNzmax(mxArray *array_ptr, int nzmax);
```



设置*array_ptr*指向的稀疏矩阵mxArray中的*nzmax*值(见上)。如果*nzmax*改变，那么向量ir、pr和pi的大小(如果它们存在)也将随着改变。用helpdesk可得更多信息。

```
int *mxGetIr(const mxArray *array_ptr);
```

返回一个包含有行数的整数向量指针，其中第一行有数字 0，*array_ptr*指向的稀疏矩阵mxArray中有非零元素。如果操作失败，返回 NULL。

```
void mxSetIr(mxArray *array_ptr, int *ir);
```

定义*array_ptr*指向的稀疏矩阵mxArray中有非零元素的行。参数 *ir*是一个整数向量指针，包含使用的行数，这些行必须按列序来存储。在 0处开始行计数。用helpdesk可得更多信息。

```
int *mxGetJc(const mxArray *array_ptr);
```

和mxGetIr相似，但是返回的整数向量指针直接表示有非零元素的列来。用helpdesk可得更多信息。

```
void mxSetJc(mxArray *array_ptr, int *jc);
```

和mxSetIr相似，但是设置直接表示有非零元素列的向量。用helpdesk可得更多信息。

```
bool mxIsSParse(const mxArray *array_ptr);
```

如果*array_ptr*指向的mxArray是稀疏矩阵类型，返回真。

下面的程序用来创建和处理字符串 mxArrays。

命令集179 C中字符串的处理

```
mxChar
```

被字符串mxArray用来存储数据元素的数据类型。

```
mxArray *mxCreateCharArray(int ndim, const int *dims);
```

和mxCreateCellArray相似，但是是创建*n*维的字符矩阵，见命令集181。

```
mxArray *mxCreateCharMatrixFromStrings(int m,char **str);
```

和mxCreateCellMatrix相似(见命令集181)，但是是用*str*指向的字符串向量创建二维字符矩阵；*m*是字符串向量中的字符串数。

```
mxArray *mxCreateString(const char *str);
```

用字符串str创建一个字符串矩阵 mxArray。如果创建成功，则返回指向这个字符串mxArray的指针；否则返回NULL。当字符串mxArray不再使用时，应用mxDestroyArray来释放所占内存；见命令集175。

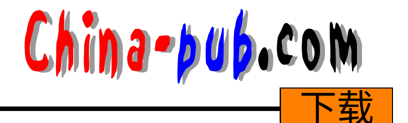
```
int mxGetString(const mxArray *array_ptr, char *buf, int buflen);
```

复制*array_ptr*指向的字符串mxArray，得到的字符串保存在 buf中。*buflen*是buf中可以存放的最大字符数。如果复制成功，返回 0；否则返回 1。

```
bool mxIsChar(const mxArray *array_ptr);
```

如果*array_ptr*指向的mxArray是字符串类型，则返回真。

MATLAB 5中一个新数据类型是多维数组；见 2.2节。用下面的程序来处理这种类型的mxArray。



注意：C中可以使用 8、16或32位的带符号或不带符号的整数和单精度浮点小数的mxArray来创建和计算。然而现在已不能在 MALTAB环境中使用它们了。

命令集180 C中多维数组的处理

```
mxArray *mxCreateNumericArray(int ndim, const int *dims, mxClassID class,
MxComplexity ComplexFlag);
```

和mxCreateCellArray相似，但是这里是创建*n*维的数字矩阵。数字类型为*class*，见命令集176中的mxClassID。如果有复数，则*ComplexFlag*设为mxCOMPLEX；否则为mxREAL。

```
void *mxGetData(const mxArray *array_ptr);
```

和mxGetPr相似，见命令集177，但是返回一个void指针。更多的是用在除双精度浮点小数以外的其他类型数字矩阵中。

```
void mxSetData(mxArray *array_ptr ,void *data_ptr);
```

和mxSetPr相似，见命令集177，但是返回一个void指针。更多的是用在除双精度浮点小数以外的其他类型数字矩阵中。

```
void *mxGetImagData(const mxArray *array_ptr);
```

和mxGetPi相似，见命令集177，但是返回一个void 指针。更多的是用在除双精度浮点小数以外的其他类型数字矩阵中。

```
void mxSetImagData(mxArray *array_ptr, void *pi);
```

和mxSetPi相似，见命令集177，但是返回一个void指针。更多的是用在除双精度浮点小数以外的其他类型数字矩阵中。

```
void mxSetLogical(mxArray *array_ptr);
```

在*array_ptr*指向的数字mxArray中设置逻辑标识符。MATLAB就会把mxArray的数据当作逻辑变量来对待，也就是0是假，其他值是真。

```
void mxClearLogical(mxArray *array_ptr);
```

去掉数字mxArray中的逻辑标识符，见上。

```
bool mxIsLogical(const mxArray *array_ptr);
```

检查数字mxArray中的逻辑标识符的设置，见上。如果设置了，就返回真；否则返回假。

MATLAB 5中一新数据类型是细胞矩阵，也称细胞数组；见 5.5节。下面的程序用来处理这种类型的mxArray：

命令集181 C中细胞矩阵的处理

```
mxArray *mxCreateCellArray(int ndim, const int *dims);
```

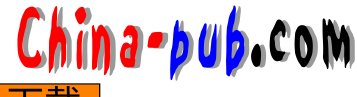
创建一个*n*维的空细胞矩阵。参数ndim是维数，*dims*是表示每维大小的向量指针。如果创建成功，就返回指向细胞矩阵的指针；否则返回 NULL或停止程序运行。

```
mxArray *mxCreateCellMatrix(int m ,int n);
```

和上个函数相似，但是是用来创建一个二维 $m \times n$的细胞矩阵。

```
mxArray *mxGetCell(const mxArray *array_ptr, int index);
```

从细胞mxArray中复制一个细胞。参数*array_ptr*是指向细胞mxArray的指针，*index*表



示第一个细胞与被复制细胞之间的细胞数；见命令集176中mxCalcSingleSubscript。如果复制成功，返回指向细胞mxArray的指针；否则返回NULL。

```
void mxSetCell(mxArray *array_ptr, int index, mxArray *value);
```

设置细胞mxArray中的一个细胞。参数*index*表示第一个细胞与被设置细胞之间的细胞数；见命令集176中mxCalcSingleSubscript。参数*value*是细胞指针，细胞的值将被设置在*array_ptr*指向的mxArray中。

```
bool mxIsCell(const mxArray *array_ptr);
```

如果*array_ptr*指向的mxArray是细胞类型，则返回真。

MATLAB 5中另一新数据类型是结构；见 12.5节。下面的程序用来处理这种类型的mxArray：

命令集182　C中结构的处理

```
mxArray *mxCreateStructArray(int ndim, const int *dims ,int nfields,const
char **field_names);
```

和mxCreateCellArray相似(见命令集181)，但是是创建*n*维的结构矩阵。参数*nfields*表示每个元素中的域数，*field_names*是字符串向量指针，表示域名。

```
mxArray *mxCreateStructMatrix(int m, int n, int nfields, const char
**field_names);
```

和上个函数相似，但是是用来创建二维 $m \times n$的结构矩阵。

```
int mxGetNumberOfFields(const mxArray *array_ptr);
```

返回*array_ptr*指向的结构mxArray中的域数。如果操作失败，则返回0。

```
mxArray *mxGetField(const mxArray *array_ptr, int index, const char
*field_name);
```

返回*array_ptr*指向的结构mxArray中一个元素的一个域值。参数*index*表示第一个元素和返回的元素之间的元素个数；见命令集176中mxCalcSingleSubscript。参数field_name表示元素中域名的字符串，如果操作成功，返回指向这个域的指针；否则返回NULL。

```
void mxSetField(mxArray *array_ptr, int index, const char *field_name, mxArray
*value);
```

和上个函数相似，但是是用指针*value*指向的值来设置域值。

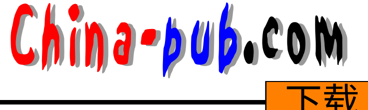
```
int mxGetFieldNumber(const mxArray *array_ptr, const char *field_name);
```

返回*array_ptr*指向的结构mxArray中一个域中的域数。字符串field_name表示域名字，如果操作成功，则返回结构中域的个数(从0开始)；否则返回－1。

```
const char *mxGetFieldNameByNumber(const mxArray *array_ptr, int
field_number);
```

返回*array_ptr*指向的结构mxArray中的域名。参数*field_number*表示结构中域的一个数字序列(从0开始)。

```
mxArray *mxGetFieldByNumber(const mxArray *array_ptr, int index, int
field_number);
```



和mxGetField相似，但是返回的是域名，*field_number*表示结构中域的一个数字序列(从0开始)。

```
void mxSetFieldByNumber(mxArray *array_ptr,int index,int field_number,
mxArray *value);
```

和mxSetField相似，但是返回的是域名，*field_number*表示结构中域的一个数字序列(从0开始)。

```
bool mxIsStruct(const mxArray *array_ptr);
```

如果*array_ptr*指向的mxArray是结构类型，则返回真。

```
int mxSetClassName(mxArray *array_ptr, const char *classname);
```

将*array_ptr*指向的MATLAB结构转换成classname指定的MATLAB对象。如果转换成功，返回0；否则返回一个非零数。当用load读入对象到MATLAB中时，要检查类classname是否存在。如果不存在，则不能将对象反转换成结构。

下面的程序用来在C中取一些特殊常数值，比如机器无穷小正数和无穷大正数。还有一些程序用来检查变量的值是否等于这些常数。

命令集183 C中特殊常数

```
double mxGetEps(void);
```

返回MATLAB中机器无穷小正数。

```
double mxGetInf(void);
```

返回MATLAB中inf的值，也就是无穷大的正数。

```
bool mxIsInf(double value);
```

如果*value*是无穷大正数，就返回真；否则返回假。

```
double; mxGetNaN(void);
```

返回MATLAB中的NaN值。

```
bool mxIsNaN(double value);
```

如果*value*是一个NaN，返回真；否则返回假。

```
bool mxIsFinite(double value);
```

如果*value*不是一个inf或NaN，就返回真；否则返回假。

下面的程序用来调试C语言程序。

命令集184 C中调试程序

```
void mxAssert(int expr, char *error_message);
```

在调试时使用，如果*expr*为假，则程序停止，并打印出*expr*、文件名、行数和**错误信息**。如果*expr*为真，则对程序没有影响。

```
void mxAssertS(int expr, char *error_message);
```

同上，但是*expr*为假时不打印出*expr*。

MATLAB 4.2中的一些程序已被新程序所代替。虽然这些旧程序**不应该**再在新的C程序中

使用，但是它们还存在，以便MATLAB 5能向下兼容。

命令集185 C中旧的矩阵程序

```
mxCreateFull   mxIsFull   mxIsString   mxFreeMatrix
```

15.2.2 C中对MAT文件的处理

下面的例子说明了如何写和读一个 MAT文件，也就是MATLAB以内部二进制格式存储的数据文件。

■ 例15.1

假设C程序中要用到一个服从正态分布的随机矩阵。这个简单的矩阵在 C程序中生成很困难，但是在 MATLAB中只需用一个命令。先定义一个 10×10的服从正态分布的矩阵，并用save命令保存：

```
OldMatrix=randn(10);        % 创建一个随机矩阵
save data OldMatrix;        % 保存这个矩阵到文件data.mat中
```

再编写一个C程序，用来读这个随机矩阵。矩阵的所有元素乘以 2，并生成一个新矩阵保存到文件**data.mat**中。C程序保存在文件**matex.c**中。

```
#include "mat.h"

void main()
{
  MATFile *mfp;
  mxArray *A_ptr, *B_ptr;
  double *A, *B;
  int M, N, i, j;
/*从文件中读矩阵*/
 mfp = matOpen("data.mat", "u");
 A_ptr = matGetArray(mfp, "OldMatrix");
 M = mxGetM(A_ptr);
 N = mxGetN(A_ptr);
 A = mxGetPr(A_ptr);

/*创建新矩阵*/
 B_ptr = mxCreateDoubleMatrix(M, N, mxREAL);
 mxSetName(B_ptr, "NewMatrix");
 B = mxGetPr(B_ptr);

/*将原矩阵乘以2后保存到新矩阵中*/
 for (i = 0; i < M; i++)
 {
   for (j = 0; j < N; j++)
   {
     B[i + M * j] = 2 * A[i + M * j];
   }
 }
```

```
/*保存新矩阵到文件中，程序结束*/
  matPutArray(mfp, B_ptr);
  matClose(mfp);
  mxDestroyArray(A_ptr);
  mxDestroyArray(B_ptr);
  exit(0);
}
```

这个程序在UNIX下进行编译(在系统提示符下输入连续的一行)：

```
gcc -ansi -I/opt/matlab52/extern/include -o matex matex.c
-L/opt/matlab52/extern/lib/sol2 -R/opt/matlab52/extern/lib/sol2
-lmat -lmx -lmi -lut
```

注意，所有的路径都和系统有关。当程序运行时，将会在文件 data.mat中添加矩阵NewMatrix，它等于2*OldMatrix。

■

下面列出了与MAT文件有关的程序，用它们可以读和写 MATLAB的二进制文件，也可以在MATLAB和C程序之间传递数据。为了使用这些程序，必须在程序中嵌入头文件 mat.h，也就是在程序开始包含下面一行：

```
#include "mat.h"
```

在读或写MAT文件之前，必须先打开这个文件，在完成之后关闭文件。用下列程序来完成打开和关闭文件。

命令集186　C中打开和关闭MAT文件

`MATFile`

二进制MAT数据文件。

`MATFile *matOpen(const char *filename, const char *mode);`

以mode方式打开文件filename，打开文件的方式有四种：“ r ”表示读方式，“ w ”表示写方式，“ u ”表示读/写方式，“ w4 ”表示写MATLAB 4 的MAT文件。如果打开成功，返回MAT文件的指针；否则返回NULL。

`FILE *matGetFp(MATFile *mfp);`

返回*mfp*指向的MAT文件的C文件指针。比如在C的函数`ferror()`中将会用到。

`int matClose(MATFile *mfp);`

关闭*mfp*指向的MAT文件。如果关闭成功，返回0；否则返回EOF。

当MAT文件处于打开状态时，就可用下面的程序来读和写文件。

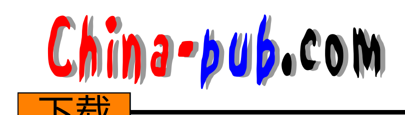
命令集187　C中读和写MAT文件

`char **matGetDir(MATFile *mfp,int *num);`

给出保存在 *mfp*指向的 MAT文件中 mxArrays的名字列表指针。参数 *num*是保存mxArrays数的变量地址。如果操作失败，返回 NULL，并且*num*变成一个负数。当不再使用mxArrays列表时，用`mxFree`释放所占内存；见命令集175。

`mxArray *matGetArray(MATFile *mfp, const char *name);`

从*mfp*指向的MAT文件中复制name指定的mxArray。如果复制成功，返回一个



mxArray指针；否则返回NULL。当不再使用时，用mxDestroyArray释放mxArray所占内存；见命令集175。

```
mxArray *matGetArrayHeader(MATFile *mfp, const char *name);
```

和上个函数相似，但是是复制数组开始部分的信息。

```
mxArray *matGetNextArray(MATFile *mfp);
```

复制*mfp*指向的MAT文件中下个mxArray。如果复制成功，返回一个mxArray指针；否则返回NULL。当不再使用时，用mxDestroyArray释放它所占的内存；见命令集175。

```
mxArray *matGetNextArrayHeader(MATFile *mfp);
```

和上个函数相似，但是是复制数组开始部分的信息。

```
int matPutArray(MATFile *mfp, const mxArray *mp);
```

将*mp*指向的mxArray写入到*mfp*指向的MAT文件中。如果文件中已有这个 mxArray，那么就被覆盖。如果写操作成功，返回0；否则返回一个非零数。

```
int matPutArrayAsGlobal(MATFile *mfp, const mxArray *mp);
```

和上个函数相似，但是当mxArray读入到MATLAB中时，它被存放到全局工作区中。在局部工作区中也可以使用它。

```
int matDeleteArray(MATFile *mfp, const char *name);
```

从*mfp*指向的MAT文件中删除name指定的mxArray。如果删除成功，返回0；否则返回一个非零数。

MATLAB 4.2中的一些函数已被一些新函数所取代，旧函数的保留是为了使 MATLAB 5能向下兼容，但是已不在C程序中使用了。

命令集188 C中与MAT文件操作有关的旧程序

matGetFull	**matGetMatrix**	**matGetNextMatrix**	**matGetString**
matPutFull	**matPutMatrix**	**matPutString**	**matDeleteMatrix**

使用下面的命令来编译和链接使用MAT文件的程序：

• UNIX：在系统提示符下输入一不间断行：

```
gcc -ansi -I/.../matlab/extern/include -o programname sourcecode.c
-L/.../matlab/extern/lib/... -R/.../matlab/extern/lib/...
-lmat -lmx -lmi -lut
```

上面路径中的三个点，...，表示这部分路径是系统安装路径，programname是用户调用的程序名，sourcecode.c是要进行编译的C源代码文件列表。这里使用的编译器是gcc，当然，其他的编译器也可以使用，只要它们按ANSI标准进行编译旧可以。如果需要，可以设置调试和优化标识；见编译器文档。

• Windows：在MATLAB提示符下输入：

```
mex sourcecode.c -f optfil
```

其中optfil表示批处理文件watengmatopts.bat(Watcom C)、msvcengmatopts.bat(Microsoft Visual C)或bccengmatopts.bat(Borland C)。参数sourcecode.c是要进行编译的C源代码文件列表。

• Macintosh：见MATLAB 5手册《应用程序接口指南》。

15.2.3 C调用MATLAB

为了使C能调用MATLAB，首先要打开一个MATLAB工程。通过调用命令engOpen就能很简单地打开一个工程；见命令集189。

下一步就是将mxArray转换成在MATLAB中可操作的形式。这可以分成两步来完成：

1) 第1步是将mxArray转换成MATLAB可理解的形式，这又有两种不同的方式。一是用程序**mxCreate**来创建矩阵，之后用**mxSetName**对它们进行命名。这些程序的描述在15.2.1节中。另一种方式是选择将一个自定义的数据结构复制到 mxArray中。然而值得注意的一点是MATLAB在存储矩阵时是按列序来保存的，而在C中是按行序来保存的，所以必须分清下标。

2) 第2步是将矩阵放入MATLAB工作区中，可以用以**engPut**开头的程序来完成；见命令集189。

现在MATLAB已准备好接收命令了。这些命令可以在普通命令窗口中给出，但是是以字符串的形式传递给函数**engEvalString**。

最后，从MATLAB到C的转换和传递也是有必要的。

这听起来相当的复杂，但是用下面的这个例子来说明就显得清楚多了。

■ **例**15.2

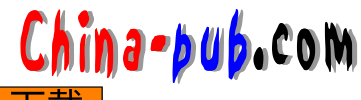
假设C程序中有一个矩阵，这个程序是有关计算机图形使用的。使用 MATLAB就能很好地将图形显示出来。编写下面C程序并将它保存在文件plotm.c中：

```
# include "engine.h"

void main()
{
  Engine *ep;
  mxArray *A_ptr;
  double* A;
  int i, j;

  /*创建一个新矩阵*/
  A_ptr = mxCreateDoubleMatrix(10, 10, mxREAL);
  mxSetName(A_ptr, "A");
 A = mxGetPr(A_ptr);
 for (i = 0; i < 10; i++)
 {
   for (j = 0; j < 10; j++)
   {
     A[i + 10 * j] = (j + 1) * (j + 1) * (i + 1) * (i + 1);
   }
 }

 /*打开一个MATLAB工程*/
 ep = engOpen("/opt/matlab52/bin/matlab");
 /*传递新矩阵*/
 engPutArray(ep, A_ptr);
 /*画出图形并保存*/
```




```
  engEvalString(ep, "mesh(A);");
  engEvalString(ep, "print picture.eps -deps;");

  /*结束*/
  engClose(ep);
  mxDestroyArray(A_ptr);
  exit(0);
}
```

在UNIX环境下编译这个程序，可以在系统提示符下输入下面一不间断行命令：

```
gcc -ansi -I/opt/matlab52/extern/include -o plotm plotm.c
-L/opt/matlab52/extern/lib/sol2 -R/opt/matlab52/extern/lib/sol2
-leng -lmat -lmx -lmi -lut
```

注意，所有的路径都和系统有关，程序运行结果如图 15-1所示。

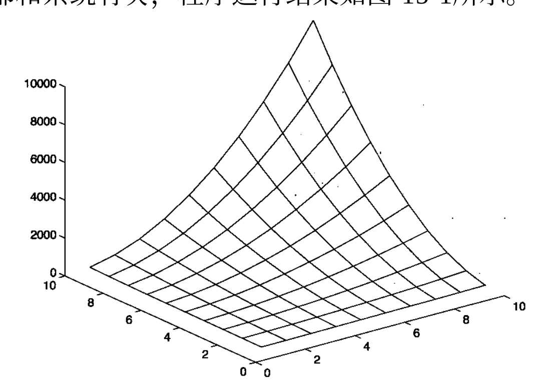


图15-1　将C程序创建的矩阵用MATLAB画出其图形

■

可以用下面的**工程程序**来处理MATLAB的调用。注意，这些程序现在已不适用于Macintosh系统。要使用这些程序，必须在程序开始包含头文件 **engine.h**，也就是在程序开始处包含下面一行：

```
#include "engine.h"
```

命令集189　C中MATLAB工程程序

```
Engine
```

表示MATLAB工程。

```
Engine *engOpen(const char *startcmd);
```

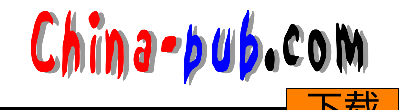
打开一个MATLAB工程，其中startcmd是一个包含打开命令的字符串，通常是“matlab”。如果打开成功，返回指向MATLAB工程的指针；否则返回NULL。

```
int engOutputBuffer(Engine *ep, char *p, int n);
```

给*ep*指向的MATLAB工程定义一个大小为*n*的文本缓冲区p，通常屏幕上显示的文本保存在这里。

```
int engEvalString(Engine *ep, const char *string);
```

在*ep*指向的MATLAB工程中对字符串string中的MATLAB命令求值。如果求值成功，



返回0；否则返回一个非零数。

```
mxArray *engGetArray(Engine *ep, const char *name);
```

在*ep*指向的MATLAB工程中从工作区复制name表示的mxArray。如果复制成功，返回指向mxArray的指针；否则返回NULL。当mxArray不再使用时，用mxDestroyArray来释放它所占的内存；见命令集175。

```
int engPutArray(Engine *ep, const mxArray *mp);
```

将*mp*指向的mxArray复制到*ep*指向的MATLAB工程的工作区中。如果复制成功，返回0；否则返回1。

```
int engClose(Engine *ep);
```

关闭*ep*指向的MATLAB工程。如果关闭成功，返回0；否则返回1。

MATLAB 4.2中的一些函数已被新函数所取代，旧函数的保留是为了使 MATLAB 5能向下兼容，但是已不在C程序中使用了。

命令集190 C中旧的MATLAB工程程序

用下面的命令来编译和链接调用 MATLAB的C程序。

```
engGetFull           enGetMatrix           engPutFull   engPutMatrix
engSetEvalCallback   engSetEvalTimeout     engWinInit
```

• UNIX：在系统提示符下输入一不间断行：

```
gcc -ansi -I/.../matlab/extern/include -o programname
sourcecode.c
-L/.../matlab52/extern/lib/...
-R/.../matlab/extern/lib/...
-leng -lmat -lmx -lmi -lut
```

上面路径中的三个点，...，表示这部分路径是系统有关，programname是用户调用的程序名，sourcecode.c是要进行编译的C源代码文件列表。这里使用的编译器是 gcc，当然，其他的编译器也可以使用，只要它们按 ANSI标准进行编译就可以。如果需要，可以设置调试和优化标识；见编译器文档。

• Windows：在MATLAB提示符下输入：

```
mex sourcecode.c -f optfil
```

其中optfil表示批处理文件watengmatopts.bat(Watcom C)、msvcengmatopts.bat(Microsoft Visual C)或 bccengmatopts.bat(Borland C)。参数sourcecode.c是要进行编译的C源代码文件列表。

15.2.4 MATLAB调用C

本节开始用一个简单的例子来说明 MATLAB对C的调用。在MATLAB中调用C和调用普通的函数文件即M文件是一样的。

MEX文件中使用的程序和矩阵格式都和 C中调用MATLAB是一样的。只不过是现在也是使用MEX文件。

文件必须有一个叫做mexFunction的主函数，mexFunction是MATLAB调用的函数。这个程序有四个参数：输入参量的个数、输出参量的个数和指向这些参量的两个指针数组。所以这个函数可以调用任何计算程序。

■ **例**15.3

创建一个MEX文件，将给出的矩阵中元素和它自己的行下标值相乘生成一个新矩阵。下面的C程序保存在文件rmult.c中：

```
# include "mex.h"

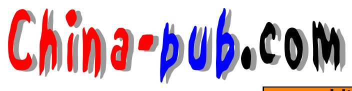

/* 完成元素和它自己的行下标相乘的程序*/
void radMult(double *Out, double *In, int M, int N)
{
  int i, j;

  for (i = 0; i <  M; i++)
  {
    for (j = 0; j < N; j++)
    {
      Out[i + M * j] = (i + 1) * In[i + M * j];
    }
  }
}

/* MATLAB调用的程序*/
void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[])
{
  mxArray *In_ptr, *Out_ptr;
  double  *In, *Out;
  int M, N;

/* 检查参量个数及其类型*/
  In_ptr = prhs[0];
  In = mxGetPr(In_ptr);
  if (nrhs != 1)
    mexErrMsgTxt("Only one input argument allowed!");
  else if (nlhs != 1)
    mexErrMsgTxt("Only one output argument allowed!");
  if ( !mxIsNumeric(In_ptr) || mxIsComplex(In_ptr) ||
        mxIsSparse(In_ptr) || !mxIsDouble(In_ptr) )
    mexErrMsgTxt
      ("Input argument must be a full floating point matrix!");

/* 创建一个新矩阵*/
  M = mxGetM(In_ptr);
  N = mxGetN(In_ptr);
```




```
    Out_ptr = mxCreateDoubleMatrix(M, N, mxREAL);
    Out = mxGetPr(Out_ptr);

  /*调用矩阵运算程序*/
    radMult(Out, In, M, N);

  /*返回新矩阵…*/
    plhs[0] = Out_ptr;
  }
```

在UNIX下编译这个程序(在系统提示符是输入)：

```
/opt/matlab52/bin/mex rmult.c
```

注意，路径和系统有关。

当程序编译完后，就可以在MATLAB提示符下输入下面内容来运行程序：

```
New =
      1      2      3
      2      4      6
      3      6      9
```

■

在下面的命令表中给出了所有MEX程序。它们可以在MEX文件中使用，也可以看作是用C编写的并已编译过的M文件。调用这些程序，必须在程序开始嵌入头文件 mex.h，也就是在程序开始处包含下面一行：

```
#include"mex.h"
```

为了使MATLAB能调用C编写的程序，必须编写一个接口函数来调用 mxFunction。从MEX文件中也可以有其他的调用，也就是用程序 mexCallMATLAB和mexEvalString来实现从C到MATLAB的调用。

命令集191 C与MATLAB的接口

```
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]);
```

是MATLAB接口函数，参数*nlhs*表示输出参量的个数，*plhs*表示指向这些参量的指针向量的指针(设置为NULL)。同样，*nrhs*表示输入参量的个数，*prhs*表示指向这些参量的指针向量的指针。输入参量不能在C程序中改变。

```
int mexCallMATLAB(int nlhs, mxArray *plhs[], int nrhs, mxArray *prhs[],
const char *command_name);
```

调用一个MATLAB函数、M文件或MEX文件。参数*nlhs*是输出参数个数(必须不大于50)，*plhs*是这些参数指针向量的指针(设置为NULL)。同样，*nrhs*是输入参数个数(必须不大于50)，*prhs*是这些参数指针向量的指针。字符串 **command_name**表示调用函数的名字，如果它是一个运算符，就用一个带单引号的字符，如‘ + ’。如果操作成功，返回0；否则返回一个非零数。当不再使用时，用 mxDestroyArray来释放*plhs*指向的mxArray所占的内存；见命令集175。

```
void mexSetTrapFlag(int trap_flag);
```

用来处理调用mexCallMATLAB时发生的错误。如果MATLAB在调用mexCallMATLAB



时发生错误，就停止运行MEX文件，并回到MATLAB提示符下。

将*trap_flag*设为0表示发生同样的错误；相反将它设为1，MEX文件恢复控制。

```
int mexEvalString(const char *command);
```

在调用工作区执行MATLAB命令**command**，不将结果传送回MEX程序。command的所有输入参数都可以在调用工作区中找到。如果操作成功，返回0；否则返回1。

```
int mexAtExit(void (*ExitFcn)(void));
```

定义一个在MEX程序退出时调用的函数。比如用来关闭文件，这个函数总是返回0。

另外在`mexFunction`中可以将数据作为参数来传递，这样就可以在MATLAB工作区中直接读和写数据。

命令集192　C和MATLAB数据交换

```
mxArray *mexGetArray(const char *name, const char *workspace);
```

从工作区复制一个mxArray，字符串**name**是mxArray的名字，**workspace**表示要复制的mxArray所在工作区，有下列字符串可用：

“base”表示在MATLAB当前工作区内搜索变量**name**。

“caller”表示在调用函数的工作区内搜索变量**name**。

“global”表示在全局变量列表中搜索变量**name**。

如果复制成功，返回复制的mxArray的指针；否则返回NULL。当它不再使用时，用`mxDestroyArray`释放所占内存；见命令集175。

```
int mexPutArray(mxArray *array_ptr, const char *workspace);
```

和上个函数相似，但是是将mxArray复制到工作区中。如果复制成功，返回0；否则返回1。

```
const mxArray *mexGetArrayPtr(const char *name, const char *workspace);
```

返回一个不同工作区中mxArray的只读指针，字符串**name**是mxArray的名字，workspace是要复制mxArray(见mexGetArray)的工作区。如果操作失败，返回NULL。注意，这只能从mxArray中读它的值，而不能改变。

下面的程序用来在MATLAB命令窗口中输出错误信息、警告和其他文本。

命令集193　C中的错误处理和打印

```
int mexPrintf(const char *format, arg1, arg2,...);
```

在MATLAB窗口中以ANSI C输出格式输出字符串。参数**format**是ANSI C格式字符串，*arg1*，*arg2*，...是输出的可选参数。

```
void mexErrMsgTxt(const char *error_msg);
```

在MATLAB窗口中输出错误信息**error_msg**，并且停止运行MEX文件。

```
void mexWarnMsgTxt(const char *warning_msg);
```

在MATLAB窗口中显示警告信息**warning_msg**，但是不停止运行MEX文件。

在MATLAB环境中，下列C程序可以完成不同的工作。例如，可以用它们来处理图形对



象；见14.2节。

命令集194　C中其他MEX程序

```
bool mexIsGlobal(const mxArray *array_ptr);
```

如果*array_ptr*指向的mxArray是全局变量，则返回真。

```
const mxArray *mexGet(double handle, const char *property);
```

用来获取MATLAB中图形对象的属性值。参数*handle*是图形对象的指针(句柄)，字符串**property**表示属性。如果操作成功，返回一个包含属性值的mxArray的指针，否则返回NULL。也可参见14.2节。

```
int mexSet(double handle, const char *property, mxArray *value);
```

和上个函数相似，但是是用来设置MATLAB中图形对象的属性。参数*value*是包含要设置的属性值的mxArray指针。如果设置成功，返回0；否则返回1。也可参见14.2节。

```
void mexAddFlops(int count);
```

将*count*加到MATLAB内部计算器上，用来计算浮点操作的次数。

下列的程序用在MEX文件中来管理内存。这些程序可以使变量在MEX程序的连续调用之间保存变量的值。

命令集195　C中内存处理

```
void mexMakeArrayPersistent(mxArray *array_ptr);
```

通常，在MEX程序退出时要对mxArray所占的内存进行释放。如果不释放所占的内存，也就是要在随后的MEX程序调用中保存mxArray的值，那么这个函数还可以使用。参数*array_ptr*表示mxArray指针。

```
void mexMakeMemoryPersistent(void *ptr);
```

和上个函数相似，但是是管理用`mxCalloc`分配的内存。参数*ptr*是指向分配内存区开始部分的指针。

```
void mexLock(void);
```

锁住MEX文件，也就是不能从内存中将它删除。

```
void mexUnlock(void);
```

对MEX文件解锁；见上。

```
bool mexIsLocked(void);
```

如果MEX文件被锁，返回真；见上。

MATLAB 4.2中的一些函数已被一些新函数所取代，旧函数的保留是为了使MATLAB 5能向下兼容，但是已不在C程序中使用了。

命令集196　C中旧的MEX程序

```
mexGetMatrix    mexGetGlobal    mexPutMatrix    mxGetMatrixPtr
mexGetFull      mexGetGlobal    mexPutMatrix    mxGetMatrixPtr
mexIsInf        mexGetNaN       mexIsNaN        mexIsFinite
```

使用下面的命令来对调用MATLAB的C程序进行编译和链接：

• **UNIX**：在系统提示符下输入一不间断行：

```
.../matlab52/bin/mex sourcecode.c
```

上面路径中的三个点，...，表示这部分路径与系统有关，programname是用户调用的程序名，**sourcecode.c**是要进行编译的C源代码文件列表。可能的话，用户要配置编译或链接环境；见MATLAB 5《应用程序接口指南》。

• **Windows**：在MATLAB提示符下输入：

```
mex sourcecode.c
```

可能的话，用户要配置编译或链接环境；见MATLAB 5《应用程序接口指南》。

• **Macintosh**：参考MATLAB 5《应用程序接口指南》来进行配置。配置完后可以在MATLAB提示符下输入：

```
Mex sourcecode.c
```

在不同的系统下创建出的MEX文件的后缀名也不一样。可以用MATLAB命令`mexext`来检查系统使用的后缀名类型。

可以对MEX程序进行调试，但是添加一个编译调试符— `g`。在UNIX环境下，编译完后，当MATLAB已运行，就可在系统提示符下输入— `Ddbx`来运行程序。在MATLAB提示符下输入`dbmex on`来运行要调试的MEX文件。然而，在MEX文件运行前，返回到调试器中来列程序、设置断点等。从MATLAB提示符下可以输入`dbmex stop`到调试器中。在调试器中可以输入`continue`返回到MATLAB中。在Windows和Macintosh中调试可以参考MATLAB 5手册《应用程序接口指南》。

15.3 MATLAB和FORTRAN

FORTRAN和C的不同之处在于MATLAB FORTRAN库只能用来处理字符串和二维矩阵、满矩阵和稀疏矩阵，而且矩阵的元素要求是双精度浮点小数。

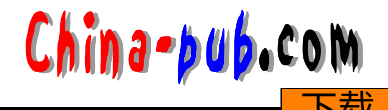
15.3.1 FORTRAN中对mxArray的操作

可以用下面表中列出的程序来对mxArray进行操作。这里的‘mxArray’可以是字符串或二维矩阵、满矩阵和稀疏矩阵，而且要求矩阵元素是双精度浮点小数。

C中的许多操作都通过指针来完成。FORTRAN中没有指针，所以操作起来很不方便。在FORTRAN中所有的平台上都用的是MATLAB的4字节的整数指针，除了在Alpha和SGI64中用的是8字节的整数指针(因此对表中的命令调用必须根据不同的平台进行相应的调整)。矩阵的所有运算都必须通过以`mxCopy`开头的转换程序来进行。

有一些编译器也支持%val结构，这是对FORTRAN 77和FORTRAN 90的一个扩充。也就是可以在调用的子程序间传递变量值，也称为‘值调用’。如果这种结构可用，就不必将指针转换成数据，也就是可以用`mxGetPr`和`mxGetPi`来返回指针；见命令集200。将%`val`代替这些指针传递到子程序中，在子程序中将这些指针声明为包含双精度浮点小数的FORTRAN矩阵；见下例15.6。参考FORTRAN编译手册，可知它是否支持%val结构。

如果要动态分配内存也要用到% val结构；见MATLAB 5手册《应用程序接口指南》和下例15.6。



命令集197 FORTRAN中指针处理

```
subroutine mxCopyPtrToInteger4(px, y, n)
integer*4 y(n)
integer*4 px,n
```

从*px*指向的MATLAB整数向量中复制*n*个整数到FORTRAN中标准整数向量y中。参数*px*可以是**ir**或**jc**向量的指针；见命令集201中mxGetIr和mxGetJc。

```
subroutine mxCopyInteger4ToPtr(y,px,n)
integer*4 y(n)
integer*4 px, n
```

从FORTRAN普通整数向量**y**中复制*n*个整数到MATLAB *px*指向的整数向量中。参数*px*可以是**ir**或**jc**向量的指针；见命令集201中mxGetIr和mxGetJc。

```
subroutine mxCopyPtrToPtrArray(px, y, n)
integer*4 y(n)
integer*4 px, n
```

复制指针*px*到FORTRAN的普通整数向量y中，参数*px*可以是**ir**或**jc**向量的指针；见命令集201中mxGetIr和mxGetJc。

```
subroutine mxCopyPtrToReal8(px, y, n)
real*8  y(n)
integer*4  px, n
```

从MATLAB中的*px*指向的浮点小数向量中将*n*个浮点小数复制到FORTRAN中的普通浮点小数向量**y**中，参数*px*可以是**pr**或**pi**向量的指针；见命令集200中mxGetPr和mxGetPi。

```
subroutine mxCopyReal8ToPtr(y, px, n)
real*8 y(n)
integer*4 px, n
```

从FORTRAN的普通浮点小数向量**y**中将*n*个浮点小数复制到MATLAB的*px*指向的浮点小数向量中，参数*px*可以是**pr**或**pi**向量的指针；见命令集200中mxGetPr和mxGetPi。

```
subroutine mxCopyPtrToComplex16(pr, pi, y, n)
complex*16 y(n)
integer*4 pr, pi, n
```

将*pr*(实数部分)和*pi*(虚数部分)指向的MATLAB向量中的*n*个复数复制到FORTRAN的普通复数向量**y**中，见命令集200中mxGetPr和mxGetPi。

```
subroutine mxCopyComplex16ToPtr(y,pr,pi,n)
complex*16 y(n)
integer*4 pr,pi , n
```

将FORTRAN的普通复数向量**y**中的*n*个复数复制到*pr*(实数部分)和*pi*(虚数部分)指向的MATLAB向量中，见命令集200中mxGetPr和mxGetPi。

```
subroutine mxCopyPtrToCharacter(px, y, n)
character*(*) y
integer*4 px, n
```

将*px*指向的MATLAB字符向量中的*n*个字符复制到FORTRAN的普通字符向量y中，见命令集202中mxGetString。

```
subroutine mxCopyCharacterToPtr(y, px, n)
character*(*) y
```

```
integer*4 px, n
```

将FORTRAN的普通字符向量**y**中的*n*个字符复制到*px*指向的MATLAB字符向量中，见命令集202中`mxGetString`。

下面表中的程序用来分配和释放内存。及时地释放不再使用的内存是一个好的编程经验。如果只用MATLAB程序来创建数据结构，就不必释放内存，因为在程序结束时 MATLAB会自动完成。为了以防万一，不管哪种情况都可用这些程序来释放内存。

命令集198　FORTRAN内存管理

```
integer*4 function mxCalloc(n, size)
integer*4 n, size
```

分配内存，参数*n*是要分配的元素的个数，*size*是每个元素的字节数。如果分配成功，返回指向分配内存的起始位置的指针；否则返回0或程序停止。当元素不再使用时，用mxFree来释放分配的内存。

```
subroutine mxFree(ptr)
integer*4 ptr
```

释放*ptr*指向的内存。

```
subroutine mxFreeMatrix(pm)
integer*4 pm
```

使用用`mxCreateFull`或`mxCreateSparse`分配的内存。参数*pm*是一个mxArray的指针。

命令集199　FORTRAN中处理mxArray的常用程序

```
character*32 function mxGetName(pm)
integer*4 pm
```

返回一个字符向量(最大32个字符)，包含*pm*指向的mxArray名字。如果操作失败，返回0。

```
subroutine mxSetName(pm, name)
integer*4 pm
character*(32) name
```

将*pm*指向的mxArray赋予名字**name**(最大32个字符)。

```
real*8 function mxGetScalar(pm)
integer*4 pm
```

返回*pm*指向的mxArray的第一个实数元素值。如果 mxArray是稀疏矩阵，则返回它的第一个非零元素值；如果是空矩阵，则返回一个不确定数。

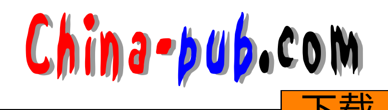
```
integer*4 function mxGetM(pm)
integer*4 pm
```

返回*pm*指向的mxArray中行数。

```
subroutine mxSetM(pm,m)
integer*4 pm, m
```

用来对*pm*指向的mxArray重构，增加/减少mxArray中的行数，参数*m*是要求的行数。如果增加/减少mxArray中的行数，则必须分配/释放内存；见`helpdesk`可得更多信息。

```
integer*4 function mxGetN(pm)
```




```
integer*4 pm
```

返回*pm*指向的mxArray的列数。

```
subroutine mxSetN(pm,n)
integer*4 pm,n
```

用来对*pm*指向的mxArray重构，增加/减少mxArray中的列数，参数*n*是要求的列数。如果增加/减少mxArray中的列数，则必须分配/释放内存；见helpdesk可得更多信息。

```
integer*4 function mxIsNumeric(pm)
integer*4 pm
```

如果*pm*指向的mxArray包含数字，则返回1，否则返回0。

```
integer*4 function mxIsDouble(pm)
integer*4 pm
```

如果*pm*指向的mxArray包含有双精度浮点小数，则返回 1；否则返回0。如果返回0，则这个mxArray不能被FORTRAN程序使用。

```
integer*4 function mxIsComplex(pm)
integer*4 pm
```

如果*pm*指向的mxArray包含有复数，则返回1；否则返回0。

下面的程序用来创建和处理二维 $m \times n$的满矩阵，矩阵的元素是双精度浮点小数。

命令集200 **FORTRAN中满矩阵的处理**

```
integer*4 function mxCreateFull(m, n, ComplexFlag)
integer*4 m, n, ComplexFlag
```

用双精度浮点小数创建一个二维 $m \times n$矩阵(mxArray)。如果矩阵中有复数，则参数*ComplexFlag*设为1；否则设为0。当矩阵不再使用时，用 mxFreeMatrix来释放矩阵所占的内存空间；见命令集 198。

```
integer*4 function mxGetPr(pm)
integer*4 pm
```

返回*pm*指向的mxArray中指向第一个实数元素的指针。如果mxArray中没有实数元素，则返回0。

```
subroutine mxSetPr(pm, pr)
integer*4 pm, pr
```

在*pm*指向的满矩阵mxArray中设置实数元素。参数*pr*是包含实数值的向量指针，这个向量必须用mxCalloc来动态分配；见命令集 198。

```
integer*4 function mxGetPi(pm)
integer*4 pm
```

和mxGetPr相似，但是是用于虚数元素。

```
subroutine mxSetPi(pm, pi)
integer*4 pm, pi
```

和mxSetPr相似，但是是用于虚数元素。

```
integer*4 function mxIsFull(pm)
integer*4 pm
```

*pm*指向的mxArray是一个满矩阵，则返回1；如果是稀疏矩阵，则返回0。

下面的程序用来创建和处理二维 $m \times n$的稀疏矩阵，矩阵的元素是双精度浮点小数。

命令集201 FORTRAN中稀疏矩阵的处理

```
integer*4 function mxCreateSparse(m, n, nzmax, ComplexFlag)
integer*4 m,n,nzmax, ComplexFlag
```

创建一个二维$m \times n$的稀疏矩阵，参数*nzmax*表示矩阵中非零元素的个数。如果矩阵中有复数元素，则参数*ComplexFlag*设为1，否则设为0。如果创建成功，返回指向矩阵的指针；否则返回0。

```
integer*4 function mxGetNzmax(pm)
integer*4 pm
```

返回*pm*指向的稀疏矩阵mxArray中非零元素的个数*nzmax*(见上)。如果发生错误，返回一个不确定数。

```
subroutine mxSetNzmax(pm, nzmax)
integer*4 pm, nzmax
```

设置*pm*指向的稀疏矩阵mxArray中非零元素的个数*nzmax*(见上)。如果改变*nzmax*值，并且如果存在向量**ir**、**pr**和**pi**，则它们也会发生改变。使用helpdesk可得更多信息。

```
integer*4 function mxGetIr(pm)
integer*4 pm
```

返回一个整数向量指针，向量中包含 *pm*指向的稀疏矩阵中有非零元素的行数，其中第一行有数字0。如果操作失败，返回0。

```
subroutine mxSetIr(pm, ir)
integer*4 pm, ir
```

定义*pm*指向的稀疏矩阵中有非零元素的行数，参数 *ir*是整数向量指针。向量中包含要使用的行数，这些行必须按列序来保存。行从0开始。使用helpdesk可得更多信息。

```
integer*4 function mxGetJc(pm)
integer*4 pm
```

和mxGetIr相似，但是返回的是直接表示有非零元素的列的整数向量指针。使用helpdesk可得更多信息。

```
subroutine mxSetJc(pm, jc)
integer*4 pm, jc
```

和mxSetIr相似，但是是用来设置有非零元素的列。使用 helpdesk可得更多信息。

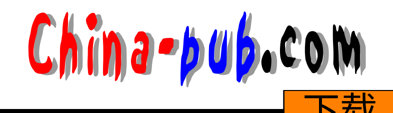
```
integer*4 function mxIsSparse(pm)
integer*4 pm
```

如果*pm*指向的矩阵是稀疏矩阵，则返回1；否则返回0。

下面的程序用来创建和处理字符串矩阵 mxArrays。

命令集202 FORTRAN中字符串的处理

```
integer*4 function mxCreateString(str)
character*(*) str
```



从字符串 **str**创建一个字符串矩阵 mxArray。如果创建成功，则返回这个字符串mxArray的指针；否则返回0。

```
integer*4 function mxGetString(pm, str, strlen)
integer*4 pm, strlen
character*(*) str
```

从*pm*指向的字符串mxArray中复制一个字符串。字符串被保存在 FORTRAN的字符串向量**str**中，*strlen*是**str**中可以保存的最大字符个数。如果复制成功，返回指向字符串mxArray的指针；否则返回0。

```
integer*4 function mxIsString(pm)
integer*4 pm
```

如果*pm*指向的mxArray是字符串矩阵，则返回1；否则返回0。

15.3.2 FORTRAN中MAT文件的处理

下面的例子给出了如何来读和写MAT文件，也就是MATLAB以二进制格式存储的数据文件。

■ **例**15.4

假设FORTRAN程序中要用到一个服从正态分布的随机矩阵。在程序中来生成相当的困难，但是在MATLAB中只需一个命令就可以创建出来。先定义一个 10×10的正态分布的矩阵，并用save命令保存：

```
OldMatrix=randn(10);      % 创建一个随机矩阵
Save data OldMatrix       % 将矩阵OldMatrix保存到文件data.mat中
```

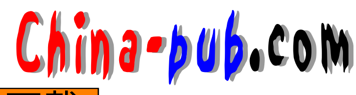
现在编写FORTRAN程序来读取这个随机矩阵且所有的元素乘以 2，并将结果作为一个新矩阵保存到文件**data.mat**中。FORTRAN程序保存在文件**matex.for**中。

```
      program main
      implicit none

      integer mfp
      integer A_ptr, B_ptr
      double precision Temp(10, 10)
      integer A, B
      integer i, j
      integer matOpen, matGetMatrix, mxCreateFull
      integer mxGetPr, matPutMatrix, stat, matClose, stat

C     Read matrix from file.
      mfp = matOpen("data.mat", "u")
      A_ptr = matGetMatrix(mfp, "OldMatrix")
      A = mxGetPr(A_ptr)
      call mxCopyPtrToReal8(A, Temp, 100)

C     Create a new matrix.
      B_ptr = mxCreateFull(10, 10, 0)
      call mxSetName(B_ptr, "NewMatrix")
      B = mxGetPr(B_ptr)
```




```
C     Set the new matrix to the old*2.
      do 100 i = 1, 10
         do 110 j = 1, 10
            Temp(i, j) = 2.0 * Temp(i, j)
110      continue
100   continue
      call mxCopyReal8ToPtr(Temp, B, 100)

C     Save the new matrix and finish.
      stat = matPutMatrix(mfp, B_ptr)
      stat = matClose(mfp)
      call mxFreeMatrix(A_ptr)
      call mxFreeMatrix(B_ptr)
      stop

      end
```

UNIX下编译这个程序(在系统提示符下输入一不间断行)：

```
f77 -I/opt/matlab52/extern/include -o matex matex.for
-L/opt/matlab52/extern/lib/sol2 -R/opt/matlab52/extern/lib/sol2
-lmat -lmx -lmi -lut
```

注意，所有的路径都和系统有关。程序运行后，在文件 data.mat中增加一个矩阵**NewMatrix**，它等于**2*OldMatrix**。

■

下面列出对MAT文件操作的程序，使用这些程序可以来读和写 MATLAB二进制文件，并且可以在MATLAB和FORTRAN程序之间传递数据。注意，这些程序不能用在Windows环境下，不过它们仍能用在C程序中。

命令集203 **FORTRAN中打开和关闭MAT文件**

```
integer*4 function matOpen(filename, mode)
integer*4 mfp
character*(*) filename, mode
```

以**mode**方式打开文件**filename**，打开的方式有："r"读方式、"w"写方式、"u"读/写方式、"w4"以MATLAB 4 MAT文件格式写。如果成功打开文件，返回MAT文件的指针*mfp*，否则返回0。

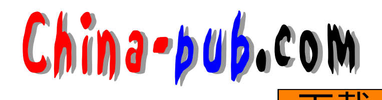
```
integer*4 function matClose(mfp)
integer*4 mfp
```

关闭*mfp*指向的MAT文件。如果成功关闭，返回0；否则返回1。

当MAT文件处于打开状态时，可以用下面的程序来写或读文件。

命令集204 **FORTRAN中MAT文件的读和写**

```
integer*4 function matGetDir(mfp, num)
integer*4 mfp, num
```



给出一个向量指针，向量包含 *mfp*指向的MAT文件中保存的 mxArray的名字。参数 *num*是存有mxArray个数的变量。如果操作失败，返回 0，并且*num*变成一个负数。当向量不再使用时`mxFree`来释放所占内存空间；见命令集 198。

```
integer*4 function matGetMatrix(mfp, name)
integer*4 mfp
character*(*) name
```

从*mfp*指向的MAT文件中复制**name**表示的mxArray。如果复制成功，返回指向mxArray的指针；否则返回0。用`mxFreeMatrix`来释放mxArray所占的内存；见命令集198。

```
integer*4 function matGetNextMatrix(mfp)
integer*4 mfp
```

从*mfp*指向的MAT文件中复制下一个 mxArray。如果复制成功，返回指向 mxArray的指针；否则返回0。用mxFreeMatrix来释放mxArray所占的内存；见命令集198。

```
integer*4 function matPutMatrix(mfp, name)
integer*4 mp, mfp
character*(*) name
```

将名为 **name**的mxArray写到*mfp*指向的MAT文件中。如果 mxArray已存在于文件中，则覆盖原来的mxArray。如果写成功，返回0；否则返回一个非零数。

```
integer*4 function matGetFull(mfp, name, m, n, pr, pi)
integer*4 mfp, m, n, pr, pi
character*(*) name
```

从*mfp*指向的MAT文件中复制名为 **name**的$m \times n$满矩阵，矩阵的元素是双精度的浮点小数。参数*pr*是矩阵实数部指针，*pi*是矩阵的虚数部指针。参数 *m*、*n*、*pr*和*pi*由函数来设置。如果复制成功，返回 0；否则返回1。当矩阵不再使用时，用`mxFree`来释放矩阵的实数部和虚数部所占的内存；见命令集 198。

```
integer*4 function matPutFull(mfp, name, m, n, pr, pi)
integer*4 mfp, m, n, pr, pi
character*(*) name
```

和上个函数相似，但是是将名为 **name**的$m \times n$满矩阵写入到MAT文件中，矩阵的元素是双精度浮点小数。如果文件中已存在这个矩阵，就覆盖原来的矩阵。

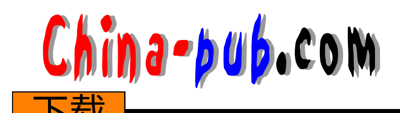
```
integer*4 function matGetString(mfp, name, str, strlen)
integer*4 mfp, strlen
character*(*) name, str
```

从*mfp*指向的MAT文件中复制最长*strlen*的字符串**str**。如果复制成功，返回 0；返回1表示**str**不是字符串；返回2表示**str**的长度大于*strlen*；返回3表示发生文件错误。

```
integer*4 function matPutString(mfp, name, str)
integer*4 mfp
character*(*) name, str
```

将字符串**str**作为字符串矩阵 **name**写入到MAT文件中。如果操作成功，返回 0；否则返回1。

```
subroutine matDeleteMatrix(mfp, name)
integer*4 mfp
```




```
character*(*) name
```

从*mfp*指向的MAT文件中删除名为的 **name**的mxArray。如果删除成功，返回 0；否则返回一个非零数。

用下面的命令来编译和链接使用 MAT文件的FORTRAN程序：

• UNIX：在系统提示符下输入一不间断行：

```
f77 -I/.../matlab/extern/include -o programname sourcecode.for
-L/.../matlab/extern/lib/... -R/.../matlab/extern/lib/...
-lmat -lmx -lmi -lut
```

上面路径中的三个点，...，表示这部分路径与系统有关，programname是用户调用的程序名，sourcecode.for是要进行编译的FORTRAN源代码文件列表。如果需要，可以设置调试和优化标识；见编译器文档。

• Macintosh：见MATLAB 5手册《应用程序接口指南》。

15.3.3 FORTRAN调用MATLAB

FORTRAN调用MATLAB，必须先启动一个 MATLAB工程。这很简单，只需调用engOpen就可以完成；见命令集205。

下一步就是将mxArray转换成在MATLAB中可操作的形式。这可以分成两步来完成：

1) 第1步是将mxArray转换成MATLAB可理解的形式。用程mxCreate开头的程序来创建一个和要传递的数据类型大小相同的矩阵 mxArray。这些程序的描述在 15.3.1中。将FORTRAN格式的数据类型复制到MATLAB格式，在C中不必这样。用mxCopy开头的程序来完成这些操作；见命令集197。

2) 第2步是将矩阵放入MATLAB工作区中，可以用程序engPutMatrix或engEvalString来完成；见命令集205。

现在MATLAB已准备好接收命令了。这些命令可以在普通命令窗口中给出，但是是以字符串的形式传递给函数engEvalString。

此后保留其他的完成变换的过程。

这听起来相当复杂，但是用下面的例子来说明就显得清楚多了。

■ 例15.5

假设FORTRAN程序中有一个矩阵，这个程序是有关计算机图形使用的。使用 MATLAB能很好地将图形显示出来。编写下面的 FORTRAN程序并将它保存在文件**plotm.for**中：

```
program main
implicit none

integer ep
integer A_ptr
integer A
integer i, j
double precision Temp(10, 10)
integer mxCreateFull, mxGetPr, engOpen
```




```
      integer engPutMatrix, engEvalString, engClose, stat

C     Create a new matrix.
      A_ptr = mxCreateFull(10, 10, 0)
      call mxSetName(A_ptr, "A")
      A = mxGetPr(A_ptr)
      do 100 i = 1, 10
         do 110 j = 1, 10
            Temp(i, j) = j * j * i * i
110      continue
100   continue
      call mxCopyReal8ToPtr(Temp, A, 100)

C     Start the MATLAB Engine.
      ep = engOpen("/opt/matlab52/bin/matlab")
C     Transfer the new matrix.
      stat = engPutMatrix(ep, A_ptr)
C     Preform the command mesh(A) and save.
      stat = engEvalString(ep, "mesh(A);")
      stat = engEvalString(ep, "print picture.eps -deps;")

C     Finish.
      stat = engClose(ep)
      call mxFreeMatrix(A_ptr)
      stop
      end
```

在UNIX环境下编译这个程序，可以在系统提示符下输入下面一不间断行命令：

```
f77 -I/opt/matlab52/extern/include -o plotm plotm.for
-L/opt/matlab52/extern/lib/sol2 -R/opt/matlab52/extern/lib/sol2
-leng -lmat -lmx -lmi -lut
```

注意，所有的路径都和系统有关。程序运行的结果如图 15-1所示。

■

可以用下面的**工程程序**来处理MATLAB的调用。注意，这些程序现在已不适用于Windows或Macintosh系统，不过在C中它们仍可用。

命令集205 FORTRAN中MATLAB工程程序

```
integer*4 function engOpen(startcmd)
integer*4 ep
character*(*) startcmd
```

启动一个MATLAB工程，字符串**startcmd**是启动命令，通常是“matlab”。如果成功，返回指向MATLAB工程的指针*ep*；否则返回0

```
subroutine engOutputBuffer(ep, p, n)
integer*4 ep, n
character*(*) p
```

给*ep*指向的MATLAB工程定义一个大小为p的文本缓冲区。通常将屏幕上显示的文件保存到这里。

```
integer*4 function engEvalString(ep, command)
integer*4 ep
character*(*) command
```

在*ep*指向的MATLAB工程中对字符串**command**中的MATLAB命令求值。如果求值成功，返回0；否则返回一个非零数。

```
integer*4 function engGetMatrix(ep,name)
integer*4 ep
character*(*) name
```

在*ep*指向的MATLAB工程中从工作区复制名为 **name**的mxArray。如果复制成功，返回指向mxArray的指针，否则返回0。当mxArray不再使用时，用`mxDestroyArray`来释放它所占的内存，见命令集198。

```
integer*4 function engPutMatrix(ep, mp)
integer*4 mp,ep
```

将*mp*指向的mxArray复制到*ep*指向的MATLAB工程的工作区中。如果在工作区中这个mxArray已存在，则覆盖它。如果复制成功，就返回0；否则返回1。

```
integer*4 function engGetFull(ep, name, m ,n, pr, pi)
integer*4 ep, m, n, pr, pi
character*(*) name
```

从*ep*指向的MATLAB工程工作区中复制名为**name**的$m \times n$满矩阵，矩阵的元素是双精度浮点小数。参数*pr*是矩阵实数部指针，*pi*是矩阵虚数部指针，参数*m*、*n*、*pr*和*pi*由函数来设置。如果复制成功，返回0；否则返回1。如果矩阵不再使用，就用mxFree来释放矩阵的实数部和虚数部所占的内存；见命令集198。

```
integer*4 function engPutFull(ep, name, m, n, pr, pi)
integer*4 ep, m, n, pr, pi
character*(*) name
```

和上个函数相似，但是是将名为**name**的$m \times n$满矩阵复制到工作区中，矩阵的元素是双精度浮点小数。如果工作区中已有这个矩阵，则覆盖它。

```
integer*4ep integer*4 function engClose(ep)
```

关闭*ep*指向的MATLAB工程。如果操作成功，返回0；否则返回1。

用下面的命令来编译和链接调用MATLAB的FORTRAN程序：

• UNIX：在系统提示符下输入一不间断行：

```
f77 -I/.../matlab/extern/include -o programname sourcecode.for
-L/.../matlab/extern/lib/... -R/.../matlab/extern/lib/...
-leng -lmat -lmx -lmi -lut
```

上面路径中的三个点，...，表示这部分路径与系统有关，**programname**是用户调用的程序名，**sourcecode.for**是要进行编译的FORTRAN源代码文件列表。如果需要，可以设置调试和优化标识；见编译器文档。

15.3.4　MATLAB调用FORTRAN

在本节开始用一个简单的例子来描述MATLAB调用FORTRAN，调用方式和在MATLAB

中调用普通函数M文件一样。

MEX文件使用的程序和矩阵格式和FORTRAN中调用MATLAB相同。然而现在也使用MEX文件。

文件中必须有一个叫做mexFunction的主函数，它是MATLAB调用的函数。这个程序有四个参数：输入参数的个数、输出参数的个数和两个指向这些参数的指针数组。所以这个函数可以调用任何计算程序。

■ **例**15.6

创建一个MEX文件，返回用给定的矩阵元素和其下标相乘得到的一个新矩阵。下面的FORTRAN程序保存在文件**rmult.for**中：

```
C     A routine multiplying the elements with their row indices.
      subroutine radMult(Out, In, M, N)
      integer M, N
C     Dynamic memory in Fortran 77!
      real*8 Out(M, N), In(M, N)

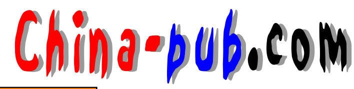
      integer i, j

      do 100 i = 1, M
         do 110 j = 1, N
            Out(i, j) = real(i * In(i, j))
110      continue
100   continue

      return
      end
C     This routine is called by MATLAB.
      subroutine mexFunction(nlhs, plhs, nrhs, prhs)
      integer nlhs, nrhs
      integer plhs(*), prhs(*)

      integer In_ptr, Out_ptr
      integer In, Out
      integer M, N
      integer mxGetPr, mxGetM, mxGetN
      integer mxCreateFull, mxIsFull, mxIsDouble

C     Check the number of arguments and their type.
      In_ptr = prhs(1)
      In = mxGetPr(In_ptr)
      if (nrhs .ne. 1) then
         call mexErrMsgTxt("Only one input argument allowed!")
      elseif(nlhs .ne. 1) then
         call mexErrMsgTxt("Only one output argument allowed!")
      endif
      if ((mxIsFull(In_ptr) .ne. 1) .or.
     $     (mxIsDouble(In_ptr) .ne. 1)) then
         call mexErrMsgTxt ("Input argument must be a full
     $          floating point matrix!")
      endif
```




```
C       Create a new matrix.
        M = mxGetM(In_ptr)
        N = mxGetN(In_ptr)
        Out_ptr = mxCreateFull(M, N, 0)
        Out = mxGetPr(Out_ptr)
C       Calls a routine that operates on the matrix.
        call radMult(%val(Out), %val(In), M, N)

C       Returns the new matrix ...
        plhs(1) = Out_ptr
        return
        end
```

在UNIX环境中编译前，必须将文件

```
/opt/matlab52/bin/mexopts.sh
```

拷贝到相同的目录下。然后编辑这个文件，在标题 sol2)下设置：

```
LDFLAGS=$"$-G -M $MATLAB/extern/lib/sol2/$MAPFILE
                   -R $MATLAB/extern/lib/sol2
                   -L $MATLAB/extern/lib/sol2
                   -leng -lmat -lmx -lmi -lut$"$
```

现在就可以用下面的命令来编译程序 (在系统提示符下输入)：

```
/opt/matlab52/bin/mex rmult.for
```

注意，路径和系统有关。

当程序编译完后，可以在MATLAB提示符下输入：

```
New = rmult([1 2 3;1 2 3;1 2 3])

New =
     1     2     3
     2     4     6
     3     6     9
```

■

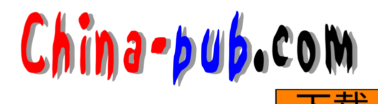
在下面的命令表中描述了所有的 MEX程序。它们可以在MEX文件中使用，并且可以看作FORTRAN编写的已编译的M文件。

为了让MATLAB能够调用FORTRAN编写的程序，必须编写一个调用 mexFunction的接口函数。从 MEX文件中，也可以反过来调用，也就是用程序 mexCallMATLAB和mexEvalString来在FORTRAN中调用MATLAB。

命令集206 FORTRAN中的MATLAB接口

```
subroutine mexFunction(nlhs, plhs, nrhs, prhs)
integer*4 nlhs, nrhs, plhs(*), prhs(*)
```

是MATLAB接口函数。参数*nlhs*是输出参数的个数，**prhs**是这些参数的指针向量。同样，*nrhs*是输入参数的个数，**prhs**是输入参数的指针向量。在 FORTRAN程序中输入参数不能改变。




```
integer*4 function mexCallMATLAB(nlhs, plhs, nrhs, prhs, name)
integer*4 nlhs, nrhs, plhs(*), prhs(*)
character*(*) name
```

调用一个MATLAB函数，M文件或MEX文件。参数*nlhs*是输出参数的个数(必须小于或等于50)，**prhs**是这些参数的指针向量。同样，*nrhs*是输入参数的个数(必须小于或等于50)，**prhs**是输入参数的指针向量。字符串**name**是被调用的函数名。如果它是一个运算符，那么它必须放在两单引号之间，如‘ + ’。如果操作成功，返回0；否则返回一个非零数。当**prhs**指向的mxArrays不再使用时，用`mxFree`来释放它们所占的内存；见命令集198。

```
integer*8 function mexCallMATLAB(nlhs, plhs, nrhs, prhs, name)
integer*4 nlhs, nrhs
integer*8 plhs(*), prhs(*)
character*(*) name
```

和上个函数相似，但是是用于Alpha和SGI64系统中。

```
subroutine mexSetTrapFlag(trap_flag)
integer*4 trap_flag
```

用来处理在使用 `mexCallMATLAB`时发生的错误。如果 MATLAB在调用`mexCallMATLAB`中有一个错误，就停止运行 MEX文件，并且退回到MATLAB提示符下。如果将*trap_flag*设为0，也会进行相同的操作。但另一方面，如果将它设为 1，那么MEX文件将再获控制。

```
integer*4 function mexEvalString(command)
character*(*) command
```

在调用工作区中执行 MATLAB命令**command**。不会将结果返传给 MEX程序。命令**command**的所有输入参数都必须可以在调用工作区中找到。如果操作成功，返回 0；否则返回一个非零数。

```
integer*4 function mexAtExit(ExitFcn)
subroutine ExitFcn( )
```

定义一个MEX程序退出前调用的函数。例如用来关闭文件的函数。总是返回 0。

在`mexFunction`中可以将数据作为参数来传递，这样就可以在 MATLAB工作区中直接读和写数据。

命令集207 FORTRAN和MATLAB之间的数据传递

```
integer*4 function mexGetMatrix(name)
character*(*) name
```

从调用工作区中拷贝名为**name**的mxArray。如果拷贝成功，返回指向这个mxArray的指针；否则返回0。当mxArray不再使用时，用`mxFree`来释放它所占的内存；见命令集198。

```
integer*4 function mexGetGlobal(name)
character*(*) name
```

和上个函数相似，但是是从MATLAB的全局工作区中拷贝mxArray。

```
integer*4 function mexPutMatrix(mp)
integer*4 mp
```

和函数mxGetMatrix相似，但是是将mxArray拷贝到工作区中。如果拷贝成功，返回0；否则返回1。

```
integer*4 function mexGetMatrixPtr(name)
character*(*) name
```

返回调用工作区中名为 **name**的mxArray指针。这使得可以在 MEX程序中读和修改mxArray。不能对这个mxArray进行释放内存操作。

```
integer*4 function mexGetFull(name, m, n, pr, pi)
integer*4 m, n, pr pi
character*(*) name
```

从调用工作区中拷贝名为**name**的$m \times n$满矩阵，矩阵的元素是双精度浮点小数。参数*pr*是矩阵实数部指针， *pi*是矩阵虚数部指针 (*m*，*n*，*pr*和*pi*由函数来设定)。如果拷贝成功，返回0；否则返回1。当矩阵不再使用时，用mxFree释放矩阵实数部和虚数部所占的内存；见命令集198。

```
integer*4 function mexPutFull(name, m, n, pr, pi)
integer*4 m, n, pr, pi
character*(*) name
```

和上个函数相似，但是是将 $m \times n$的满矩阵拷贝到调用工作区中，矩阵的元素是双精度浮点小数。

下面的程序用来在FORTRAN中获取特殊常数值，如机器无穷小数和正无穷大数。还有用来检查变量值是否等于这些常数的函数。

命令集208 FORTRAN**中的特殊常数**

```
real*8 function mexGetEps( )
```

返回MATLAB中机器无穷小数。

```
real*8 function mexGetInf( )
```

返回MATLAB中inf值，也就是无穷大数。

```
integer*4 function mexIsInf(value)
real*8 value
```

如果*value*等于inf，返回1；否则返回0。

```
real*8 function mexGetNaN( )
```

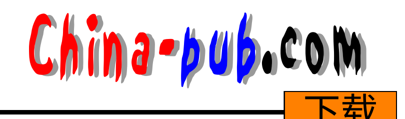
返回MATLAB中NaN的值。

```
integer*4 function mexIsNaN(value)
real*8 value
```

如果**value**等于NaN，返回1；否则返回0。

```
integer*4 function mexIsFinite(value)
real*8 value
```

如果*value*不等于inf或NaN，返回1；否则返回0。



下面的程序用来在MATLAB命令窗口中输出错误信息和其他文本内容：

命令集209 FORTRAN中错误处理和打印

```
subroutine mexPrintf(format, arg1, arg2,...)
character*(*) format, arg1, arg2,...
```

在MATLAB窗口中输出ANSIC输出格式的字符串。参数**format**是ANSIC格式字符串，*arg*1，*arg*2，...是输出的可选参数。

```
subroutine mexErrMsgTxt(error_msg)
character*(*) error_msg
```

在MATLAB窗口中输出错误信息**error_msg**，并且停止运行MEX文件。

用下面的命令来编译和链接调用 MATLAB的FORTRAN程序：

• UNIX：在系统提示符下输入一不间断行：

```
.../matlab52/bin/mex sourcecode.for
```

上面路径中的三个点，...，表示这部分路径与系统有关， programname是用户调用的程序名，**sourcecode.for**是要进行编译的FORTRAN源代码文件列表。如果可能用户可以配置编译或链接环境；参见MATLAB 5手册《应用程序接口指南》和例15.6。

• Windows：在MATLAB提示符下输入：

```
mex sourcecode.for
```

如果可能，用户可以配置编译或链接环境；参见 MATLAB 5手册《应用程序接口指南》和例15.6。

• Macintosh：参见MATLAB 5手册《应用程序接口指南》来配置，然后在 MATLAB提示符下输入如下命令：

```
mex sourcecode.for
```

创建的MEX文件将会有一个系统决定的后缀名，可以借助 MATLAB命令mexext来检查系统的后缀名类型。

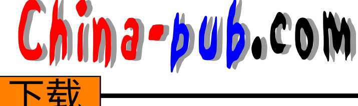
可以对MEX程序进行调试，但是添加一个编译调试符— g。在UNIX环境下，编译完后，当MATLAB已运行，就可在系统提示符下输入— `Ddbx`来运行程序。在MATLAB提示符下输入`dbmex on`来运行要调试的MEX文件。然而，在MEX—文件运行前，返回到调试器中来列程序、设置断点等。在MATLAB提示符下输入`dbmex stop`，可以到调试器中；在调试器中输入`continue`，可以返回到MATLAB中。在Windows和Macintosh中调试可以参考MATLAB 5手册《应用程序接口指南》。

15.4 MATLAB和高级文件管理

MATLAB中的下面这些命令对文件管理很有用：

命令集210 文件名

`fullfile(dir1, dir2,fname)`	连接一个字符串。字符串 **dir1**, **dir2**, ...(目录名)表示文件**fname**的路径。返回的字符串表示带有完整路径的文件**fname**。MATLAB会在目录**dir1**，**dir2**，...之间插入一个目



	录分隔符。分隔符由系统来决定，比如UNIX下用/，PC上用\。
`[path,name,ext,ver]=fileparts(file)`	在变量**path**、**name**和**ext**中返回文件的路径、文件名和扩展名。在VMS中，变量**ver**包含文件的版本号。
`filesep`	返回系统使用的目录分隔符。

有时在MATLAB中要用二进制文件，所以如果另一个程序需要或生成某种特定格式的文件，那么就必须能对各种格式的文件进行读和写。以下面的例子开始。

■ **例**15.7

假设要在文件中存储一个Hadamard(哈达马德)矩阵，这可以用在2.8节讨论过的命令`save`来完成。然而在使用命令`save`和`load`时不能控制矩阵的的格式。使用MATLAB中低级文件处理命令可以控制文件的格式和精度。

编写如下的代码，将一个64×64的哈达马德矩阵保存到文件**hada.mtl**中：

```
fp        = fopen('hada.mtl','w');
antok     = fwrite(fp,hadamard(64),'int8');
[msg,err] = ferror(fp);

if err ~= 0
  disp('An error occurred when writing to the file:')
  disp(msg)
end

err = fclose(fp);

if err ~= 0
  disp('Could not close the file.')
end
```

■

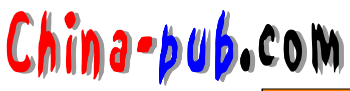
从这个例子中可以看出，文件必须先以写模式打开，然后要关闭文件。在使用命令`save`和`load`时就不必进行这些操作。

MATLAB中用命令`fopen`和`fclose`来打开和关闭文件。

命令集211 打开和关闭二进制文件

`fopen(filename,op)` 打开名为字符串**filename**的文件，返回一个文件指针。如果发生错误，返回－1。字符串**op**表示文件的打开模式，可以有下列模式：

'r'	只读模式。
'r+'	读写模式。
'w'	覆盖已存在的文件。或若文件不存在，则创建新文件。只读模式。
'w+'	覆盖已存在的文件。或若文件不存在，则创建新文件。读写模式。



	‘a’ 追加已存在的文件。或若文件不存在，则创建新文件。只写模式。
	‘a+’ 追加已存在的文件。或若文件不存在，则创建新文件。读写模式。
	字母‘r’、‘w’、和‘a’分别代表读、写和追加。 PC和VMS用户必须区别二进制文件和文本文件。在字符串 **op**中加一个字母‘t’来区别；见help fopen。
`[fp,msg]=fopen(filename,op,ark)`	同上。如果发生错误，*fp*被赋予值－1，字符串**msg**保存的是错误信息，用来解释错误信息，参考有关 C语言的手册。字符串**arch**规定数据存储的机器格式；输入help fopen可得更多信息。
`[filename,op,ark]=fopen(fp)`	返回文件*fp*的有关信息。字符串**filename**是文件的名字，字符串**op**是文件的打开模式，字符串**arch**是文件打开的机器格式。
`fclose(fp)`	关闭文件*fp*。如果关闭失败，返回－1；否则返回0。
`fclose(‘all’)`	关闭所有打开的文件。如果关闭失败，返回－1，否则返回0。

■ **例**15.8

(a) 以读和追加模式打开文件**hada.mtl**：

```
[fp,msg] = fopen('hada.mtl','a+');

if fp == -1
  disp(msg)
end
[f,op,ark] = fopen(fp)
err         = fclose(fp)

if err ~= 0
  disp('Could not close the file.')
end

f =
      hada.mtl

op =
      a+

ark =
      ieee-be

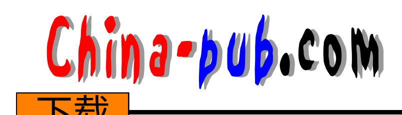
err =
      0
```

(b) 关闭(a)中的文件：

```
err = fclose(fp)

err = 0
```

■



下面的两个命令fwrite和fread用来对二进制文件进行写和读操作。

命令集212 写和读二进制文件

fwrite(fp,A,prec)	将矩阵**A**的列写入到文件*fp*中。函数返回写入元素的个数，字符串**prec**规定使用的精度，有20多种不同的精度可用。输入help fwrite可得更多信息。第4个参数可用来在矩阵A的元素之间写空字节。
fread(fp)	从文件*fp*中读并返回数据。
[A,c]=fread(fp,s,prec)	从文件*fp*中读出数据到矩阵**A**中。矩阵**A**的大小由*s*来决定： s=*n*　一个长度为*n*的列向量。 s=inf　文件*fp*中所有的数据读入到一个列向量中。 s=[*m*, *n*]　从文件*fp*中的数据以列向量方式读入到 $m\times n$ 矩阵中。 标量*c*是从文件中无错误地读出的元素个数。
feop(fp)	如果到*fp*所指的文件结尾，返回1；否则返回0。

■ **例15.9**

下面对在例15.7中创建的文件hada.mtl进行读操作：

```
clear;

fp = fopen('hada.mtl','r');
A  = fread(fp,[64,64],'int8');

fclose(fp);
whos
nnz(A-hadamard(64))      % Just checking...
```

结果显示：

```
 Name        Size        Elements     Bytes     Density    Complex

    A      64 by 64         4096      32768       Full        No
  ans       1 by 1             1          8       Full        No
   fp       1 by 1             1          8       Full        No

Grand total is 4098 elements using 32784 bytes

ans =
      0
```

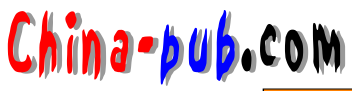
这和写入到文件中的哈达马德矩阵相同。

■

MATLAB可以创建和读带格式的文本文件。在这些操作中， MATLAB以列向量的方式读和写矩阵的元素。然而MATLAB是以行向量的方式来读和写文本文件，所以有必要进行转换。

命令集213 写和读带格式的文本文件

fprintf(fp,fstr,A,...)	依据字符串fstr给出的格式，将矩阵或数组**A**的元



	素写入到文件*fp*中。字符串**fstr**包含的是象C程序语言中的格式字符。表15-1给出了最常用的代码列表，可见C语言手册或输入`help fprintf`，可得更多信息。函数最后返回写入元素的个数。
`fprintf(fstr,A,...)`	将格式数据输出到屏幕上。
`[A,c]=` `fscanf(fp,fstr,s)`	从文件*fp*中读出数据到矩阵**A**中。如果*s*是标量，那么**A**就是一个列向量。如果s=[*m*, *n*]，那么**A**就是一个从*fp*中以列向量方式读出的$m \times n$矩阵。字符串**fstr**规定数据的读出格式，和fprinft使用的格式字符相同。
`fgetl(fp)`	从文件*fp*中读出下一行到字符串中，函数返回这个字符串。
`fgets(fp)`	从文件*fp*中读出包含eol(行结束)字符的下一行到字符串中，函数返回这个字符串。

在命令`fprintf`和`fscanf`以及5.1.2节中命令`sprintf`和`sscanf`的格式字符串中使用的字符列在表15-1中。

表15-1 命令FPRINTF和SPRINTF中使用的格式符

控制字符	格式符
\n 换行	%e 科学计数格式,小写e
\r 回车	%E 科学计数格式,大写E
\b 退格	%f 十进制格式
\t 横向跳格	%s 字符串
\f 新页	%u 整型数
” 单引号 ’	%i 同上类型
\\ 反斜杠	%X 十六进制,大写
\a 响铃	%x 十六进制,小写

除了这些外，还可以规定数据宽度、小数的位数和对齐方式。通常，MATLAB使用的是右对齐方式，但是在百分号和格式符之间的负号表示左对齐方式。数据的宽度和小数的位数也可以在百分号和格式符之间来规定，如%[－][#.#]F，括号内的部分是可选的，F是格式符；见表15-1。

■ **例**15.10

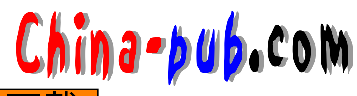
(a) 用命令sprintf来输出不同格式的数值*p*：

```
twodec = sprintf('%4.2f',pi)    % 两位小数

twodec =
        3.14
ninedec = ...
    sprintf('The number pi = %11.9f',pi)    % 九位小数

ninedec =
          The number pi = 3.141592654

scfform = sprintf('%E',pi)     % 科学计数法
```




```
scfform =
          3.141593E+00
```

(b) 下面的程序将函数$f(x)=1/x$在区间[0, 4]上的值以格式表形式写入到文件tab.txt中。

```
% 写函数f(x) = 1/x 的格式表
% 左列是占四位的整数
% 右列占六位，且有三位小数

x = 1:4; Y = zeros(4,2);

Y(:,1) = x';%                  % Y的第1列
Y(:,2) = 1 ./ x';              % Y的第2列

fp = fopen('tab.txt','w+');    % 打开文件tab.txt

fprintf(fp,'%4.0f \t %6.3f \n',Y');
fclose(fp);
```

当程序运行时，会生成文件**tab.txt**。可以用`type tab.txt`检查文件内容。

```
1       1.000
2       0.500
3       0.333
4       0.250
```

(c) 读(b)中创建的文件：

```
fp      = fopen('tab.txt','r');
[Tab,c] = fscanf(fp,'%f %f',[2,4]);

fclose(fp);
Tab = Tab'

Tab =
     1.0000     1.0000
     2.0000     0.5000
     3.0000     0.3330
     4.0000     0.2500
```

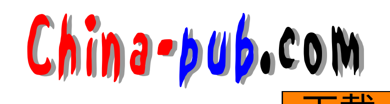
注意，读取的矩阵必须进行转置，因为MATLAB以列方式创建矩阵，而文件是以行方式来读取的。

■

在每个文件操作中，都可能会发生错误。在每个文件操作之后，都应该进行错误处理，但在例15.7中没有进行错误处理。如果在最后一个文件操作中发生错误，可以通过调用产生错误代码和错误信息的函数`ferror`来进行错误处理。

命令集214 错误信息

`msg=ferror(fp)`	如果在对文件*fp*的I/O操作过程中发生错误，返回一个错误信息。
`[msg,errn]=ferror(fp,`	返回上个命令中的错误信息，而且返回一个在C



'clear')	语言参考手册中可查的错误代码。如果给出 ' clear ', 则清空错误信息缓冲区。

■ **例**15.11

在每个文件操作后都应该进行错误检查：

```
fp          = fopen('tab.txt','r');
[Tab,c]     = fscanf(fp,'%f %f',[3,4]);
[msg,errn]  = ferror(fp);

if errn ~= 0
  disp('An error occurred when reading from tab.txt!')
  disp(msg)
end

fclose(fp);
Tab = Tab'
```

如果对例15.10(b)中创建的文件**tab.txt**进行操作，将会给出：

```
An error occurred when reading from tab.txt!
At end-of-file.

Tab =
    1.0000    1.0000    2.0000
    0.5000    3.0000    0.3330
    4.0000    0.2500         0
```

■

有三个函数可以来控制文件指针 *fp*的位置：`fseek`、`ftell`和`frewind`。可以在文件内移动文件指针，检查文件指针的位置。

命令集215 文件指针的位置

`frewind(fp)`	将文件指针 *fp*移动到文件开始处。
`fseek(fp,nb,u)`	移动文件指针到 *u*和*nb*指定的位置。这里的 *nb*是距离 *u*规定的开始位置的字节数： −1 文件起始处。 0 当前位置。 1 文件结尾。 如果 *nb*<0 从*u*处向文件起始处移动指针 *nb*个字节。 *nb*=0 移动指针到 *u*处。 *nb*>0 从*u*处向文件结尾处移动指针 *nb*个字节。
`ftell(fp)`	将从文件起始处读取的字节数作为文件指针的位置返回。负数表示已有错误发生。



15.5　和其他程序的结合

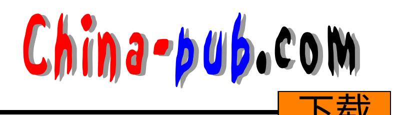
和其他程序结合使用 MATLAB的可能性将由机器决定，可阅读手册和请教系统管理员。在本节中提到一些例子。

在Lotus 1-2-3的电子数据表格 WK1中和 ASCII电子数据表格中可以读和写文件。MATLAB 5.2中有命令来读和写hdf文件和写vrml文件。

命令集216　读/写特殊文件格式

`dlmread(fname,st,r,c,v)`	从文件**fname**中读一个ASCII电子数据表格。表格中单元有分隔符 st，例如‘\t’表示制表符。如果给出*r*和*c*，表示从 *r*行和 *c*列处开始读取表格单元(数字10开始计数)。如果给出向量v=[*r*1, *c*1, *r*2, *c*2]，表示读取(r1, c1)和(*r*2 ,*c*2)之间的部分表格单元。(*r*1, *c*1)表示左上单元，(*r*2, *c*2)表示右下单元。
`dlmwrite(fname,A,st,r,c)`	将MATLAB矩阵**A**转换成一个ASCII电子数据表格保存到文件 **fname**中。表格中单元要有分隔符 st。如果给定 *r*和*c*，就从 *r*行和 *c*列的开始处写入矩阵**A**(数字从0开始计数)。
`wklread(fname,r,c,v)`	从文件**fname**中读取Lotus 1-2-3 WK1电子数据表格。如果给定*r*和*c*，表示从*r*行和*c*列处开始读表格单元数据(数字从0开始)。如果给定的*v*是向量[*r*1, *c*1, *r*2, *c*2]，表示读取(*r*1, *c*1)和(*r*2, *c*2)之间的部分表格单元。(*r*1 *c*1)表示左上单元，(*r*2 *c*2)表示右下单元。也可以以字符串形式给出区间v，如‘`a1...c5`’或‘`sales`’。
`wklwrite(fname,A,st,r,c)`	将MATLAB矩阵A转换成一个Lotus 1-2-3 WK1电子数据表格保存到文件 **fname**中。如果给定 *r*和*c*，就从*r*行和*c*列的开始处写入矩阵**A**(数字从0开始)。
`hdf`	对hdf格式文件进行读和写操作。使用 help hdf可得更多信息。为了能在 MATLAB中使用函数对hdf文件进行操作，必须熟悉 hdf库；相关信息可在 `http://hdf.ncsa.uiuc.edu`上找到。也可参见命令集162中的命令`imread`和`imwrite`。
`vrml(h,filename)`	写一个vrml 2.0文件，文件中包含有句柄*h*的图形对象和它的子对象。用字符串 `filename`得到文件，如果没有给出后缀名，就加上后缀名 `.wrl`。如果没有给出文件名，就使用`matlab.wrl`作为文件名。

如果MATLAB运行在Microsoft Windows环境下，可以使用 DDE(动态数据交换)来和其他程序交换数据。这可以用在 MS Word和MATLAB之间进行数据交换。对 MATLAB Notebook Suite有访问权的用户可以在 Word中交互地写他们自己的 MATLAB命令，代码中可以混合有



MATLAB产生的无格式文本和图形。而且，可以通过矩阵编辑器来处理数组等。

在UNIX网络中，可以在其他计算机上启动 MATLAB工程，这样就可以使用计算机上的空闲资源来完成计算，而在自己本地计算机上有图形和命令窗口。

MATLAB有许多的工具箱，这里提到其中几个重要的工具箱：

- The Signal Processing Toolbox　　用于信号处理。
- The Optimization Toolbox　　用于优化。
- The Symbolic Math Toolbox　　通过链接Maple V用于符号数学。

有关这些和其他工具箱的更多信息可以输入 `expo`，然后选择 TOOLBOXES 来得到；也可参见附录C。总之，它们可以描述成M文件的集合，每个集合都用于某个特定的领域。

MATLAB结合其他程序使用还存在很大的可能性，在将来的 MATLAB版本中会进行进一步的开发。



附录A MATLAB初步

这是一个MATLAB的简短介绍。建议在阅读本书的同时使用介绍的命令。若要详细地了解有关命令，请参见每一节的内容和 /或使用*help*或*helpdesk*。所有的命令见附录D和命令列表。

A.1 启动和退出MATLAB

根据使用的计算机，单击图标或输入 matlab就可以启动MATLAB。详细内容见2.1节。以下信息出现在MATLAB的命令窗口中：

```
                    < M A T L A B (R) >
          (c) Copyright 1984-98 The MathWorks, Inc.
                    All Rights Reserved
                    Version 5.2.0.3084
                       Jan 17 1998


To get started, type one of these: helpwin, helpdesk, or demo.
For product information, type tour or visit www.mathworks.com.
```

使用这些命令是非常好的。命令 tour和demo也是非常有实用价值的。在 MATLAB提示符后键入一个命令，当按下回车键后，就执行该命令。

```
>> tour ↩
```

在本书的其他节将不给出 MATLAB提示符，因为这将使读者阅读起来比较困难。命令应在提示符>>后键入，但提示符不在每一行中列出。

注意，本书使用不同的字体以区别用户在提示符处输入的命令和 MATLAB返回的运行结果。见1.2节。

退出MATLAB，只需输入quit并回车即可。从这以后将不强调在命令行后输入回车键。

```
>>quit
```

如果想要终止MATLAB的运行，就要同时按下‘CTRL’和‘c’键。MATLAB将停止其运行的所有工作，并且在屏幕上给出提示符，等待输入。

```
>>
```

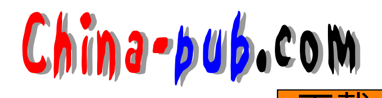
A.2 基本赋值和计算

通常，MATLAB能被当作计算器使用：

```
5011 + 13

ans =
        5024
```

同一行上可以有多条命令：




```
2^5, 2*(3+2)

ans =
    32

ans =
    10
```

通常，变量用于保存所赋的值和结果。如果没有赋值， MATLAB将结果存放在名为*ans*的变量中。现在定义变量并赋值：

```
x = 14

x =
    14

y = 3*x
y =
    42
```

所有的基本数学函数在MATLAB中有定义(见2.4节)：

```
sin(x)

ans =
    0.9906
```

圆括号’()’，可在数学式子中使用。

```
u = 2*x - y;
w = 2*(x-y);
exp((2-u)/(w-2))

ans =
    0.7589
```

注意，在命令行尾的分号‘；’是MATLAB‘quietly’执行赋值命令，即在屏幕上不回显信息，但计算照常执行。

MATLAB中的变量通常为向量或矩阵：

```
vcol = [1;2;3;4], vrow = [5 6 7 8]

vcol =
     1
     2
     3
     4

vrow =
     5     6     7     8

A = [1 2 3 4;5 6 7 8;9 10 11 12]
```

```
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

注意，各行要用分号隔开。

在单个命令中，函数可用于向量或矩阵。

```
sqrt(vcol)

ans =
    1.0000
    1.4142
    1.7321
    2.0000
```

到现在为止，已经定义了许多变量。输入以下命令，可以得到变量列表：

```
who

Your variables are:

A          u          vrow       x
ans        vcol       w          y
```

命令whos也将显示当前的变量，同时还显示出每个变量的其他信息。试使用这个命令，并看看如何区别变量是标量还是向量。

MATLAB在程序运行过程中保存所有的变量，清除变量应输入：

```
Clear
```

先前的变量现在全被清除。此时，如果输入 who，将不会返回任何信息；见 2.3节。向量可通过使用元素操作运算符来生成；见 4.3节。

```
vector = 0:8

vector =
     0     1     2     3     4     5     6     7     8

vector2 = 0:0.5:2

vector2 =
         0    0.5000    1.0000    1.5000    2.0000
```

命令linspace和logspace也可用来创建向量；见 4.3节。通过使用双操作符向量也可直接计算。

```
values = 2.^vector

values =
     1     2     4     8    16    32    64   128   256
```

注意，先前使用的运算符 ^，表示应对向量中的每一个元素进行操作。见 3.5节。

借助箭头键，能重复先前所给的命令；见 2.1节。如果输入有误，它就能避免再写过长的

表达式，这样能节省很多时间。

将向量或矩阵放入括号中能定义一个新的表达式，但是大小必须匹配：

```
Table = [vector;vector.^2;vector.^3]

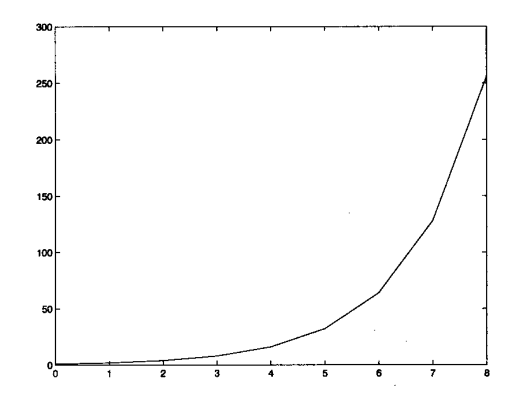
Table =
       0     1     2     3     4     5     6     7     8
       0     1     4     9    16    25    36    49    64
       0     1     8    27    64   125   216   343   512
```

A.3 简单图形

MATLAB对于处理简单图形和高级图形都是一种非常好的工具。在 MATLAB中画图形，其数据必须存储在向量或矩阵中。例如，画出 2的乘幂的图形有三步。首先，创建一个有值的向量。第二，用这些值对函数求值。第三，画出向量图形；见 13.1节。因为已经在A.2节中创建了变量vector 和values，所以就能直接使用以下命令来画图形：

```
plot(vector, values)
```

结果是一个简单图，其刻度是自动给定的；见图 A-1。



图A-1 函数2^x的分段线性图形

为了使图形更加光滑，应使用更多的值，例如使用 0～8之间的100个数值。linspace命令对于创建长向量是非常有效的，输入：

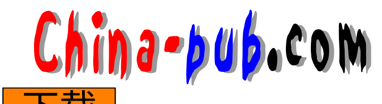
```
vector = linspace(0,8,100);
values = 2.^vector;
clf;
plot(vector,values);
```

用命令clf在画图之前清除图形窗口。现在试着使用不同的线型。给出命令 plot，及额外的参数‘：’或‘+’。这些能结合颜色；例如，用兰色的点划曲线：

```
plot(vector, values,'b-.');
```

其他线型和颜色可以在13.1节，特别是在表13-1中找到。

三维图形可用相同的方法画出。例如，使用 surf和mesh命令，生成数据放入矩阵中并画出图形。当在画带有两个变量的函数图形时，生成所需的数据可稍微灵活些。这些值或网



格点在平面上是离散的点。生成这些网格最好的方法是使用命令 meshgrid，参见13.4节。

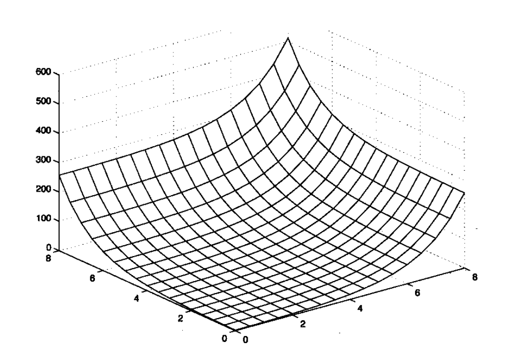
```
vector2 = 0:0.5:8;
[X,Y] = meshgrid(vector2);
mesh(X,Y,2.^X+2.^Y);
```

括号‘ [] ’用于接收多个返回参数。分号‘ ; ’用来控制结果的回显。如果是大矩阵，就有必要记住这些。给出的网格图形如图 A-2所示。

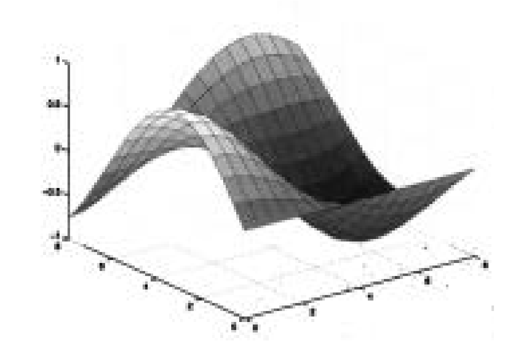
曲面图形和网状图形非常相似，画出的是实际的曲面而不是网格线，见图 A-3。曲面图形是用命令surf生成的。

```
surf(X,Y,cos(X./2).*sin(Y./2));
```

结果如图A-3所示。



图A-2　函数2^x+2^y的网状图形



图A-3　使用surf 命令生成函数cos(x/2)sin(y/2)的图形曲面

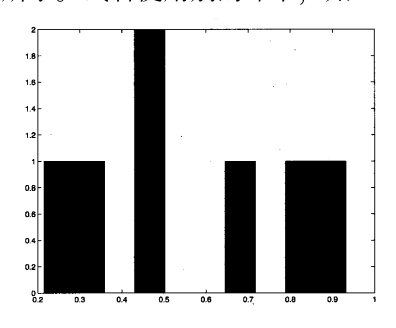
画统计图表应使用命令 hist，见6.5节。举例说明，用命令 rand生成元素值在区间 [0, 1]中的随机向量，rand命令见4.1节。

```
random = rand(1,7)

random =
    0.4553 0.3495 0.4523 0.8089 0.9317 0.6516 0.2152

hist(random);
```

hist给出的结果如图A-4所示。试着使用别的命令，如stairs和bar，见6.5节。



图A-4　频数统计直方图

A.4 线性系统和矩阵特征值

定义矩阵：

$$\mathbf{A}=\begin{pmatrix}3&4&5\\5&2&2\\1&2&3\end{pmatrix}\qquad \mathbf{B}=\begin{pmatrix}1&2&3\\1&1&1\end{pmatrix}$$

用下面的语句：

```
A = [3 4 5;5 2 2;1 2 3];
B = [1 2 3;1 1 1];
```

用命令B*A来做矩阵相乘，结果如下：

```
ans =
    16      14      18
     9       8      10
```

但是A*B的结果为：

```
??? Error using ==> *
Inner matrix dimensions must agree.
```

因为这种矩阵的乘法没有定义，见3.2节。

计算A的行列式值，可使用命令det:

```
det(A)

ans =
    -6
```

B的行列式值没有定义。也可试着对同一矩阵使用 trace、null、orth和inv命令，这些命令在7.1节有定义。

下面来看看线性方程组：

$$\begin{aligned}3x_1+4x_2+5x_3&=25\\5x_1+2x_2+2x_3&=18\\x_1+2x_2+3x_3&=13\end{aligned}$$

或者是矩阵表达式Ax=b，A是上面定义的，b如下：

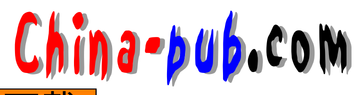
$$\mathbf{b}=\begin{pmatrix}25\\18\\13\end{pmatrix}$$

可用反斜杠运算符‘\’来求解方程组。

```
b = [25;18;13];
A\b

ans =
    2.0000
    1.0000
    3.0000
```

可以用同样的方法来解超定方程组。因为通常这些方程组没有真解， MATLAB用最小二



乘法来解。见7.2和7.7节。

特征值和特征向量可由命令eig求得。例如，对上面的矩阵**A**求：

```
[EigenVectors,EigenValues] = eig(A)

EigenVectors =
   -0.7111   -0.4501   -0.0210
   -0.6185    0.8459   -0.7756
   -0.3342   -0.2863    0.6309

EigenValues =
    8.8291         0         0
         0   -1.3373         0
         0         0    0.5082
```

矩阵**EigenVectors**的列就是**A**的特征向量，矩阵**EigenValues**的对角线上的元素就是A的特征值；见8.1节。

A.5 曲线拟合及多项式

多项式可由系数矩阵来表示；见10.1节。多项式：

$p(x)=2x^3+x^2+5x+17$

可用向量**p**=(2 1 5 17)来表示，而且可用polyval命令对任何值进行求值：

```
p = [2 1 5 17];
polyval(p,0), polyval(p,2)

ans =
    17

ans =
    47
```

用命令polyder可对多项式进行微分，并且用命令conv对其进行乘法运算。

```
pprim = polyder(p)

pprim =
     6     2     5
```

这代表$p'(x)=6x^2+2x+5$。

```
psquare = conv(p,p)

psquare =
     4     4    21    78    59   170   289
```

这代表$p(x)^2=4x^6+4x^5+21x^4+78x^3+59x^2+170x+289$。

下面在同一个图中用三个子图画出这三个多项式。命令 subplot见13.3节。

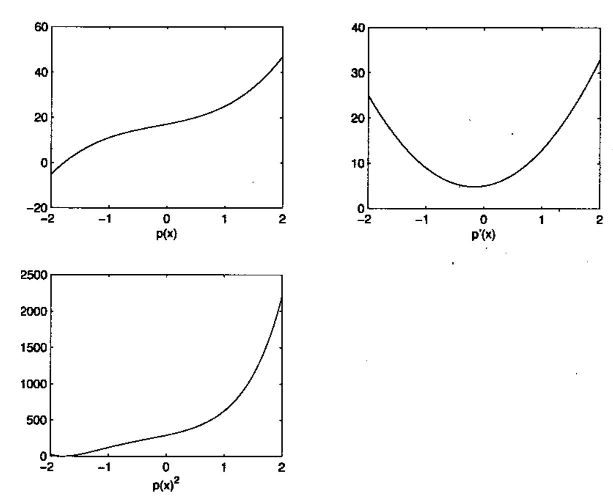
```
x = linspace(-2,2,50);
```

```
clf;
subplot(2,2,1);
plot(x,polyval(p,x));
xlabel('p(x)');

subplot(2,2,2);
plot(x,polyval(pprim,x));
xlabel('p''(x)');

subplot(2,2,3);
plot(x,polyval(psquare,x));
xlabel('p(x)^2');
```

命令xlabel可在当前图形中x轴下方写一字符串；如图 A-5所示。



图A-5 多项式$p(x)=2x^3+x^2+5x+17$及其微分和多项式$p(x)^2$的图形

多项式可用来拟合数据曲线。假设以下数据在一次实验中获得，并且输入到 MATLAB中：

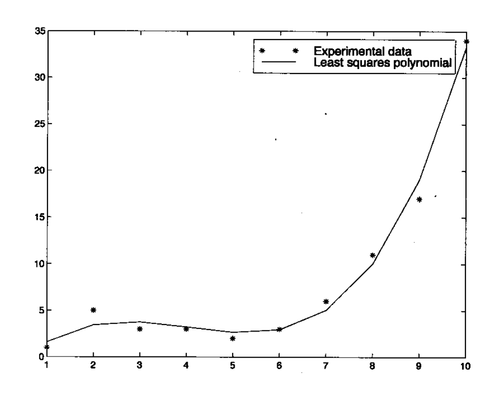
```
x = 1:10;
y = [1 5 3 3 2 3 6 11 17 34];
```

命令polyfit可返回多项式，也就是多项式的系数，该多项式表示对指定的次数拟合最好的最小二乘曲线；见 10.4节。例如此例，最好的 4次多项式已经找到。同时，原始数据点也在多项式的同一图中画出。

结果如图 A-6所示。

```
clf;
ypol = polyfit(x,y,4);
plot(x,y,'*',x,polyval(ypol,x),'b-');
legend('Experimental data','Least squares polynomial');
```

注意，有些向量可用相同的 plot命令画出彼此的连线图形。命令 legend给出一个含有对不同向量表示图解释的工具包。此命令定义在 13.3节中。对不同次数的多项式和不同的数据集合使用polyfit和polyval命令。



图A-6　数据集合和拟合的4次多项式

内插值法和外插值法可用不同的 interp命令来实现。若想对 x=4.3在上面的数据集合中内插求值，可输入：

```
interp1(x,y,4.3)

ans =
    2.7000
```

这是线性内插值法，也可看成是‘向上查阅表’。当然，其他可能性也是存在的，输入 help interp来看看。内插值法定义在10.4节中。

A.6　简单的程序

一段程序是MATLAB语句的序列，它们由条件语句和循环语句来控制。在 M文件中方便地就能保存这些命令。其中M文件是后缀名为**.m**的文本文件。

下面来看看保存在**draw.m**文件中的MATLAB语句：

```
% 绘制函数图形的程序
% 在%标记后做注释

% 显示解释文本

disp('This program plots f(x) in the interval [a,b]');

% 从用户读入数据

ftext = input('Give a function: ','s');
a     = input('Give lower bound a: ');
b     = input('Give upper bound b: ');

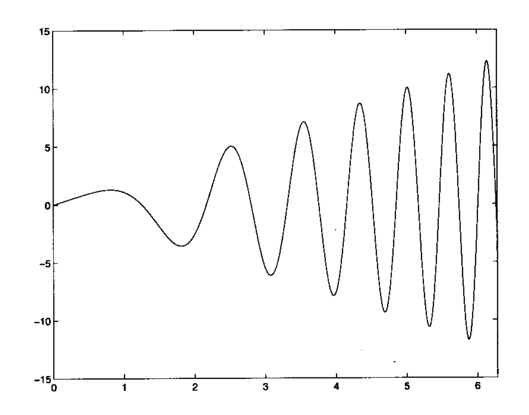
clf; fplot(ftext,[a b]);        % 绘制该函数图形
```

在MATLAB提示符后输入draw便可运行该程序。例如下列情况：

```
This program plots f(x) in the interval [a,b].
Give a function: 2.*x.*cos(x.^2)
Give lower bound a:  0
```

```
Give upper bound b:  2.*pi
```

根据上面的输入，程序运行的结果如图 A-7所示。



图A-7 用M文件draw绘制函数$2x\cos x^2$的图形

程序draw.m是一个命令文件的例子。还有函数文件，它的后缀名也必须是 .m，并且在第一行有关键字function。现在举个例子，下面的函数保存在 M文件div.m中：

```
function y = div(n,d);

% 该函数用来计算整数的除法
% 因此 n= rem(n, d)+y*d

y = (n-rem(n,d))/d;
```

该函数可以作为MATLAB中的一个函数来被调用：

```
div(1234,7), check = 1234/7

ans =
    176

check =
  176.2857
```

有关M文件的信息见2.9 和12.3节。

A.7 函数分析

可用命令fmin来寻找单变量函数的局部最小值，对于多变量函数可用命令 fmins；见10.3节。

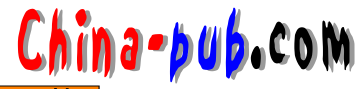
求函数的最小值：

$$f(x) = \frac{x - 1.96}{x^2 + 1.15}$$

首先要创建一个计算函数$f(x)$值的文件。该文件取名为f.m。

```
function y = f(x)

y = (x-1.96)./(x.^2+1.15);
```



命令：

```
fmin('f',-1,1)
```

给出的结果如下：

```
ans =
   -0.2742
```

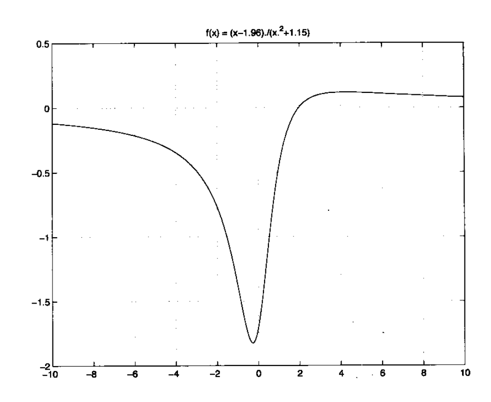
用下面给出的命令画出函数图形，就能知道所得的结果是否正确：

```
fplot('f',[-10 10]);
title('f(x) = (x-1.96)./(x.^2+1.15)');
grid;
```

title命令可给出图形的标题，此命令在13.3节中介绍。命令grid可给图形加上网格，它的介绍也在13.3节中。也能找到0值的准确位置，即对$f(x)=0$成立的x值。这是用命令fzero来求的，它需要一个迭代的初始值(见10.2节)。在图A-8中，可以看到2接近于$f(x)=0$，于是就输入：

```
fzero('f',2)

ans =
     1.9600
```



图A-8　用fplot绘制的M文件函数图形

A.8　积分

可以使用11.1节中介绍的命令quad来计算定积分。计算：

$$\int_0^{\pi} \frac{\sin(x)}{x}\,dx$$

必须在名为sinx_x.m的M文件中定义函数$\sin(x)/x$：

```
function y = sinx_x(x)

y = sin(x)./x;
```

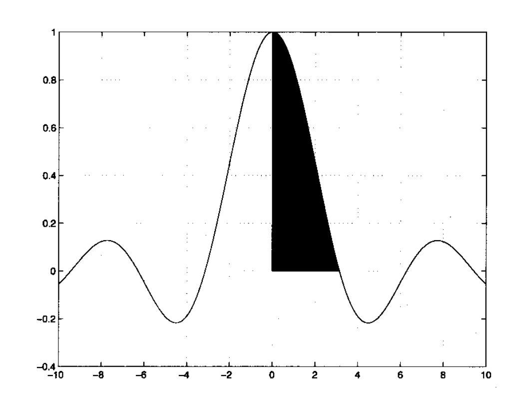
现在只需输入：

```
quad('sinx_x',1E-8,pi)
```

给出的结果为：

```
ans=
1.8520
```

注意，该积分不能被解析。函数图形如图 A-9所示。



图A-9　在区间$-10 \le x \le 10$内函数$\sin(x)/x$的图形

标记的区间为积分值。这个区域使用了定义在 14.2节中的图形对象 patch来填充。首先用fplot命令画出函数sinx_x的图形，并加上网格。

```
fplot('sinx_x',[-10 10]); grid on;
```

现在给出patch命令，并且保留一个‘句柄’，以便稍后能改变补片的属性。

```
h = patch([0.001 0.001:0.01:pi pi], ...
    [0 sinx_x(0.001:0.01:pi) 0], 'r' )

h =
   73.0005
```

patch的第一个参数定义了图形各边的x坐标。第二个参数定义了y坐标。最后一个参数定义了画图使用的颜色，这里的‘r’表示红色。现在可以输入下面的命令来改变补片的边颜色：

```
set(h,'EdgeColor',[0 0 1]);
```

要想查看由set命令设置的其他属性，可输入get(h)。这些命令在14.2节中介绍。

A.9　普通微分方程

假设现在有一个存放在文件xprim.m中的函数xprim，并定义如下：

```
function y = xprim(x,t)

y = (x-1.96)./(x.^3+1.15);
```

除了x，还有t作为输入参数，虽然函数值并不与 t有关。这只是因为要求解下面的常微分方程：

$$\frac{dx}{dt} = \frac{x - 1.96}{x^3 + 1.15}$$

用定义在11.2节中的MATLAB命令ode45来解。

先在区间$0 \le t \le 10$内、初始值为$x(0)=1$来求解(*)，调用ode45命令如下：




```
[x1,t1]=ode45(‘xprim’,[0 10],1);
```

结果是两个向量**x1**和**t1**，并绘制出它们的图形：

```
clf;
plot(x1,t1);
```

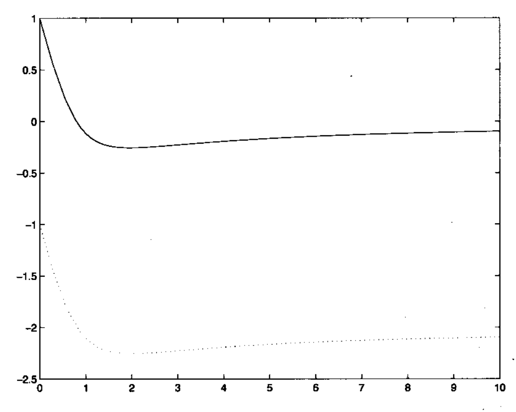
现在用相同的t区间、不同的初始值$x(0)=-1$来求解方程(*)：

```
[x2,t2]=ode45(’xprim’,[0 10],-1);
```

将这个函数图形绘制在同一个图中，这样就要在绘制新的向量图形之前给出`hold on`命令：

```
hold on;
plot(x2,t2,’:’);
hold off;
```

结果如图A-10所示。



图A-10　不同初始值的常微分方程的解的图形



附录B　线性代数中的定义和基本概念

这是对线性代数和矩阵代数基础的一个概要， MATLAB中也包含了用到的所有概念。

B.1　向量

线性空间由可以进行加和数乘运算的向量组成。

线性空间R^n由列向量组成：

$$\mathbf{x}=\begin{pmatrix}x_1\\x_2\\\vdots\\x_n\end{pmatrix}\quad \mathbf{y}=\begin{pmatrix}y_1\\y_2\\\vdots\\y_n\end{pmatrix}$$

其中，元素x_k和y_k为实数，长度为n。

在线性空间C^n中，元素可以为复数。

加法的定义是各元素分别相加：

$$\mathbf{x}+\mathbf{y}=\begin{pmatrix}x_1+y_1\\x_2+y_2\\\vdots\\x_n+y_n\end{pmatrix}$$

数乘定义为各元素分别与数α相乘：

$$\alpha\mathbf{x}=\begin{pmatrix}\alpha x_1\\\alpha x_2\\\vdots\\\alpha x_n\end{pmatrix}$$

所有元素均为零的向量定义为零向量。

$$\mathbf{0}=\begin{pmatrix}0\\0\\\vdots\\0\end{pmatrix}$$

在线性空间的p个向量中，即$\mathbf{x}_1$，$\mathbf{x}_2$，……$\mathbf{x}_p$的集合，如果至少有一个向量可以由其他向量线性表示，则称这p个向量是线性相关的。

$$\mathbf{x}_p=\alpha_1\mathbf{x}_1+\cdots+\alpha_{p-1}\mathbf{x}_{p-1}$$

这里α_i为标量。

如果不能这样表示，则称这些向量线性无关。线性无关最通常的定义是： $\alpha_1\mathbf{x}_1+\alpha_2\mathbf{x}_2+\cdots\cdots+\alpha_p\mathbf{x}_p=0$成立，当且仅当$\alpha_1=\alpha_2=\cdots\cdots=\alpha_p=0$。

■ **例**B.1

向量

$$\mathbf{x}_1=\begin{pmatrix}1\\0\\0\end{pmatrix}\quad \mathbf{x}_2=\begin{pmatrix}1\\1\\0\end{pmatrix}\quad \mathbf{x}_3=\begin{pmatrix}1\\1\\1\end{pmatrix}$$

在线性空间R^n中是线性无关的。

■

线性空间中线性无关向量的最大个数称为线性空间的维数。 R^n和C^n的维数均为n。注意：在有些情况下认为C^n是$2n$维的更为方便，这样就能分成实部和虚部两部分。

线性空间的基指的是一些向量的集合，这个空间中所有的向量都能由这些向量线性表示。基中向量的个数等于空间的维数。线性空间中有无穷多组基。

■ **例**B.2

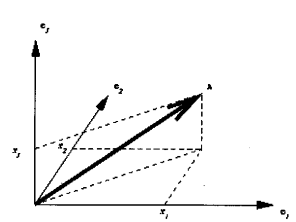
在例B.1中的向量形成R^3和C^3空间中的基。向量：

$$\mathbf{e}_1=\begin{pmatrix}1\\0\\0\end{pmatrix}\quad \mathbf{e}_2=\begin{pmatrix}0\\1\\0\end{pmatrix}\quad \mathbf{e}_3=\begin{pmatrix}0\\0\\1\end{pmatrix}$$

形成同样空间中更常用的基，有：

$$\mathbf{x}=\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}=x_1\mathbf{e}_1+x_2\mathbf{e}_2+x_3\mathbf{e}_3$$

这是R^3空间中一个由基向量线性表示的任意向量。可用图 B-1来表示说明。



图B-1 向量和它的分量

■

C^n中两个向量**x**和**y**的内积或点积，通常写作(**x**, **y**)或<**x**, **y**>，定义为：

$$(\mathbf{x},\mathbf{y})=\sum_{i=1}^{n}\bar{x}_i y_i$$

如果严格限在R^n空间中，则x_i的复数共轭将是不必要的。可以使用下一节将要介绍的符号，$(x, y)=x^H y$。

C^n中向量的欧几里德范数$\|x\|_2$定义为：

$$\|\mathbf{x}\|_2^2=(\mathbf{x},\mathbf{x})=\sum_{i=1}^{n}|x_i|^2=\mathbf{x}^H\mathbf{x}$$

范数用来度量向量的大小或长度。还有许多其他范数，将在B.6节中介绍其中的一些范数。

如果(x, y)=0，则称两个向量**x**和**y**正交。

两个向量x和y之间的角度θ是按下式来定义的：

$$\cos\theta=\frac{(\mathbf{x},\mathbf{y})}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2}$$

已经知道两个正交向量之间的夹角是 $\pi/2$或90度，即两个向量是垂直的。零向量与任何向量都正交。

如果非零向量集合 $\mathbf{x}_1$，$\mathbf{x}_2$，⋯，$\mathbf{x}_p$中所有向量都正交，则它们构成正交系，其中的向量也是线性无关的。因此，正交化比线性无关的条件更强。如果 $\mathbf{x}_1$，$\mathbf{x}_2$，⋯，$\mathbf{x}_p$形成一个正交系，并且每个向量的欧氏范数均为1，则称为标准正交系。标准正交系中的向量有如下关系：

$$(\mathbf{x}_j, \mathbf{x}_k) = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases}$$

■ **例**B.3

例B.2中的向量 $\mathbf{e}_1$，$\mathbf{e}_2$，$\mathbf{e}_3$构成R^n(和C^n)中的标准正交系。在标准的笛卡儿坐标系中，它们分别代表x, y, z轴。

■

除了以列向量的形式定义外，还可以以行向量的形式定义上述所有概念。

$$\mathbf{v} = (v_1, v_2, \ldots, v_p)$$

但是，使用列向量有几个优点。

B.2 矩阵介绍

矩阵是一个以行列形式排列的数字矩形数组。一个有 m行n列的矩阵称为$m \times n$矩阵。例如，这里有一个2×3矩阵：

$$\begin{pmatrix} 2 & 1 & 3 \\ 3 & -2 & 0 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$$

矩阵中的数字称为矩阵的元素或分量。如果矩阵命名为 **A**，矩阵**A**的元素称为α_{ij}，这里i代表行下标，j代表列下标，即α_{ij}代表i行j列的元素。

$n \times n$矩阵称为方阵。

矩阵的大小由行数m和列数n给出。对于方阵来说，n有时也指矩阵的阶数。

矩阵中从左上角到右下角的对角线称为主对角线，主对角线上的元素称为对角元素 a_{ii}。从右上角到左下角的对角线称为反对角线。主对角线上方和下方的对角线分别称为上对角线和下对角线。

大小相同的两个矩阵相加定义为矩阵的各个元素分别相加。矩阵 **C**=**A**+**B**，也就是元素c_{ij}= a_{ij}+ b_{ij}。

数乘的定义也是每个元素分别相乘。矩阵 αA的元素为αa_{ij}。

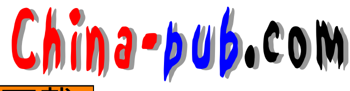
矩阵乘法仅在左侧矩阵的列数等于右侧矩阵的行数时才有意义。矩阵 **C**=**AB**是一个$m \times n$矩阵，这里**A**为$m \times p$矩阵，**B**为$p \times n$矩阵。

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$$

元素c_{ij}为**A**中i行和**B**中j列的内积。

即使**AB**有意义，但**BA**不一定有意义。如果**A**和**B**都是n阶方阵，那么**AB**和**BA**将都有意义，但是通常**AB**≠**BA**。矩阵乘法是不可交换的。

n阶单位矩阵是一个$n \times n$矩阵，其中除对角线上元素为1外，其余元素均为0，用I或I_n表示。**A**矩阵乘单位矩阵，结果不变，因此有 **IA**=**A**和**AI**=**A**。



列向量可看成是 $n\times 1$ 矩阵，而行向量可看成是 $1\times n$ 矩阵。有时将一个标量看成 1×1 矩阵也是十分有用的。

如果x是一个有 n 个分量的列向量，而 $\mathbf{A}$ 是一个 $n\times n$ 阶矩阵，那么 $\mathbf{Ax}$ 也是有 n 个分量的列向量。这称为**矩阵—向量乘法**。

转置是将矩阵的行和列交换位置，转置运算符记做 T。如果 $\mathbf{A}$ 是一个元素为 a_{ij} 的 $m\times n$ 矩阵，那么转置矩阵 $\mathbf{A}^T$ 是一个元素为 a_{ji} 的 $n\times m$ 矩阵。转置也可以看成是这样： A的第1列作为转置矩阵中的第1行， $\mathbf{A}$ 的第2列作为转置矩阵中的第2行，依次类推。

矩阵的共轭是一个矩阵，其中的元素是原矩阵中复数元素的共轭。结果记做 $\bar{\mathbf{A}}$ 。一个常用的操作符是共轭转置，这将形成矩阵 $\bar{\mathbf{A}}^T$ 或等价的 $\overline{\mathbf{A}^T}$ 。该矩阵通常记做 $\mathbf{A}^H$， $\mathbf{A}^*$，在MATLAB中记做 $\mathbf{A}'$。

两个列向量 $\mathbf{x}$ 和 $\mathbf{y}$ 的内积常写成：

$$(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \bar{x}_i y_i = \mathbf{x}^H \mathbf{y}$$

欧氏范数可写成 $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^H\mathbf{x}}$ 。注意： $\mathbf{x}^H\mathbf{y}$ 是一个标量，因为它是 $1\times n$ 阶矩阵和 $n\times 1$ 阶矩阵的内积。相反， $\mathbf{xy}^H$ 是一个 $n\times n$ 阶矩阵。

B.3　矩阵概念

矩阵不只是一个数字的集合。一些重要而有用的数学概念都与矩阵有关。

矩阵 $\mathbf{A}$ 的秩， $rank(\mathbf{A})$ 是矩阵 $\mathbf{A}$ 中线性无关列的列数，并且总是等于矩阵 $\mathbf{A}$ 中线性无关行的行数。如果 $\mathbf{A}$ 为 $m\times n$ 矩阵，则秩小于或等于 $\min(m, n)$。

方阵 $\mathbf{A}$ 的行列式， $\det(\mathbf{A})$，是一个可以用不同方式定义和计算的标量。有下列结论：

1) $\det(\mathbf{A}) = \det(\mathbf{A}^T)$.

2) $\det(\mathbf{A}^H) = \overline{\det(\mathbf{A})}$.

3) 如果 $\mathbf{A}$ 中有两行相等，或某一行可由其他行线性表示，则 $\det(\mathbf{A})=0$。对于 $\mathbf{A}$ 的列也有同样的结论。

4) 某行减去另一行与一个标量的乘积，行列式不变。对于列也有同样的结论。

5) 交换任意两行，行列式变号。对于列也有同样的结论。

6) 主对角线下方所有元素均为零的矩阵称为上三角矩阵，其行列式为主对角元素的乘积。对于下三角矩阵也有同样的结论。

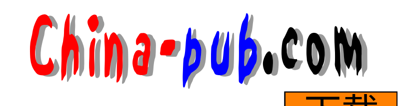
7) 矩阵乘积的行列式等于行列式的乘积。这是一个重要的乘法定理： $\det(\mathbf{AB})=\det(\mathbf{A})\det(\mathbf{B})$。

8) 矩阵行列式的计算可用高斯消元法来很好地求得。

n 阶线性方程组可以记成如下明确的形式：

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \qquad\qquad\vdots \\ a_{n1}x_1 + a_{2n}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

或用 $\mathbf{A}=(a_{ij})$， $\mathbf{x}=(x_1, x_2, \cdots, x_n)^T$ 和 $\mathbf{b}=(b_1, b_2,\cdots, b_n)^T$ 来表示：



Ax=b

将向量a_1, a_2,⋯, a_n看作**A**的列，该方程组可被写成如下的半压缩形式。

$$x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_n\mathbf{a}_n = \mathbf{b}$$

当且仅当det(**A**)≠0时方程组有唯一解。

$m \times n$阶矩阵**A**的值域$\mathcal{R}$ (**A**)是**A**的列a_1, a_2,⋯, a_n的所有线性的组合。这是一个线性空间，并且$\mathcal{R}$ (**A**)的维数等于*rank*(**A**)。

矩阵**A**的零空间$\mathcal{N}$ (**A**)是所有使得**Ax=0**的向量集合，即齐次方程组的解。这也是一个线性空间，并且维数等于*m*—*rank*(**A**)。

A^T的值域和零空间的定义同上。

方程系**AX=I**是一个矩阵方程，其中**A**是一个$n \times n$矩阵，而I为n阶单位阵。使用符号$\mathbf{x}_1$, $\mathbf{x}_2$,⋯, $\mathbf{x}_n$和$\mathbf{e}_1$=(1, 0, ⋯, 0)^T, $\mathbf{e}_2$=(0, 1, ⋯, 0)^T, ⋯, $\mathbf{e}_n$=(0, 0, ⋯, 1)^{T}作为**X**和**I**的列，根据：

$$\mathbf{X} = (\mathbf{x}_1 \quad \mathbf{x}_2 \quad \ldots \quad \mathbf{x}_n) \qquad \mathbf{I} = (\mathbf{e}_1 \quad \mathbf{e}_2 \quad \ldots \quad \mathbf{e}_n)$$

可以将矩阵方程写成线性方程组的集合：

$$\mathbf{A}\mathbf{x}_k = \mathbf{e}_k \quad k = 1, 2, \ldots, n$$

当且仅当det(**A**)≠0时方程组有唯一的解。**AX=I**的解**X**被称为**A**的逆，记为$\mathbf{A}^{-1}$，

有$\mathbf{A}\ \mathbf{A}^{-1}=\mathbf{I}$和$\mathbf{A}^{-1}\mathbf{A}=\mathbf{I}$。

逆的计算通常也使用高斯消元法。存在逆的矩阵称为非奇异矩阵，否则称为奇异矩阵。

方阵的特征值和特征向量可用如下的方程定义：

Ax=lx

这等价于齐次方程组：

(**A**—λ**I**)**x**=0

对于每个矩阵**A**和λ，向量**x=0**都是一个解。但是，如果det(**A**—λ**I**)≠0，则方程组还有非零解。这些x_k≠**0**的解称为**A**的特征向量，相应的λ_k称为特征值或特征根。在复平面上**A**总有n个特征值λ_1, λ_2,⋯, λ_n。特征值和它相应的特征向量称为特征对。

函数ϕ(λ)=det(**A**-λI)是λ的n次多项式，也称为**A**的特征多项式。特征方程为ϕ(λ)=0。

总有这样的结论：矩阵多项式φ(**A**)=0，这就是**Cayley-Hamilton定理**。

如果**C**为非奇异矩阵，**A**和**B**定义成$\mathbf{B}=\mathbf{C}^{-1}\mathbf{A}\mathbf{C}$，则称**A**和B为相似矩阵，**A**和**B**之间的变换称为相似变换。相似变换不改变矩阵的特征值。

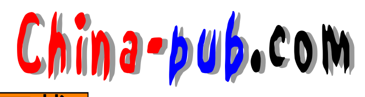
矩阵**A**的谱半径ρ(**A**)定义为$\max_i|\lambda_i|$。

如果矩阵**A**、**B**的逆存在，则对逆、转置和共轭转置如下的式子成立：

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \quad \text{如果所有的逆都存在}$$
$$(\mathbf{AB})^{T} = \mathbf{B}^{T}\mathbf{A}^{T}$$
$$(\mathbf{AB})^{H} = \mathbf{B}^{H}\mathbf{A}^{H}$$

下面的等价链包含了上述的大部分定义。**A**为n阶方阵。

线性方程**A x=b**有唯一的解
⇔
det(A≠0
⇔



A^{-1}存在
⇔
A的秩为n
⇔
A的列线性无关
⇔
A的行线性无关
⇔
A的值域维数为n
⇔
A的零空间维数为0
⇔
齐次方程组$Ax=0$有唯一的零解
⇔
λ=0不是A的特征值
⇔

B.4　矩阵分类

矩阵可用多种方式分类。一个矩阵中如果所有非对角元素均为零，则称为对角阵。如果所有主对角线下方的元素均为0，则称为上三角阵，如果对角线上的元素也都为零，则称为严格上三角阵。同样，也可以定义下三角阵和严格下三角阵。

如果仅在主对角线、第一条上对角线和第一条下对角线上有非零元素，则称矩阵为三对角矩阵。更一般的是，如果所有非零元素均在主对角线周围的带形区域内，则称为带状矩阵。

如果第一条子对角线下方的元素均为零，则称为上海森伯格形式。

如果矩阵中大部分元素都是零，称为稀疏矩阵；否则，称为满矩阵。带状矩阵属于稀疏矩阵。

使用分块矩阵也是十分有用的，分块矩阵即矩阵的元素也是矩阵。实际上代数运算不变。

假设**A**，**B**的定义如下：

$$\mathbf{A}=\begin{pmatrix}\mathbf{A}_{11} & \mathbf{A}_{12}\\ \mathbf{A}_{21} & \mathbf{A}_{22}\end{pmatrix}\quad \mathbf{B}=\begin{pmatrix}\mathbf{B}_{11} & \mathbf{B}_{12}\\ \mathbf{B}_{21} & \mathbf{B}_{22}\end{pmatrix}$$

可给出$\mathbf{C}=\mathbf{AB}$：

$$\mathbf{C}=\begin{pmatrix}\mathbf{C}_{11} & \mathbf{C}_{12}\\ \mathbf{C}_{21} & \mathbf{C}_{22}\end{pmatrix}$$

其中，比如：$\mathbf{C}_{11}=\mathbf{A}_{11}\mathbf{B}_{11}+\mathbf{A}_{12}\mathbf{B}_{21}$。

块矩阵有时也称为分区矩阵。在前面的例子中，子矩阵的大小必须一致。类似块对角阵，块上三角阵的定义就不再赘述了。

矩阵还可以按数学性质分类。已介绍过的一些概念，这里再重复一下。

如果$\det(\mathbf{A})\neq 0$，则称矩阵**A**为非奇异矩阵，这意味着所有的特征值都是非零值。如果

det(A)=0，矩阵为奇异矩阵，至少有一个特征值为零。

如果$\mathbf{A}^H=\mathbf{A}$，矩阵**A**为Hermitian矩阵。这与实数阵的对称矩阵类似。特征值为实数，特征向量形成一个标准正交基。

如果$\mathbf{A}^H=-\mathbf{A}$，矩阵**A**为反Hermitian矩阵。这与实数阵的反对称矩阵类似。特征值为虚数，特征向量形成一个标准正交基。

如果$\mathbf{A}^H\ \mathbf{A}=\mathbf{A}\ \mathbf{A}^H$，则称**A**为标准阵。特征向量形成标准正交基。

如果$\mathbf{A}^H\mathbf{A}=\mathbf{I}$，则称**A**为酉矩阵。这类似于实数矩阵的正交阵。由此得 $A^{-1}=A^H$。A的列形成标准正交基，行也一样。特征值一定为1，特征向量形成标准正交基。

如果对每个$\mathbf{x}\neq\mathbf{0}$，都有$\mathbf{x}^H\mathbf{A}\mathbf{x}>0$，则称Hermitian矩阵为正定的。所有特征值均为正数。

用同样的方式可以对较弱的条件$\mathbf{x}^H\ \mathbf{A}\mathbf{x}\geq\mathbf{0}$定义半正定矩阵，特征值均为非负。

对于某个整数p，如果$\mathbf{A}^p=\mathbf{0}$，则称矩阵A为幂零矩阵。如果$\mathbf{A}^2=\mathbf{A}$，则称矩阵**A**为幂等矩阵。

如果存在相似变换**C**，使得$\mathbf{C}^{-1}\mathbf{A}\mathbf{C}$为对角阵，则称**A**是可对角化的。**A**可对角化当且仅当**A**有n个线性无关的特征向量。

B.5 特殊矩阵

所有元素均为零的矩阵称为零矩阵。所有元素均为 1的矩阵称为1矩阵。单位矩阵主对角线上元素均为1，其他元素均为零。随机矩阵的元素都是随机的。

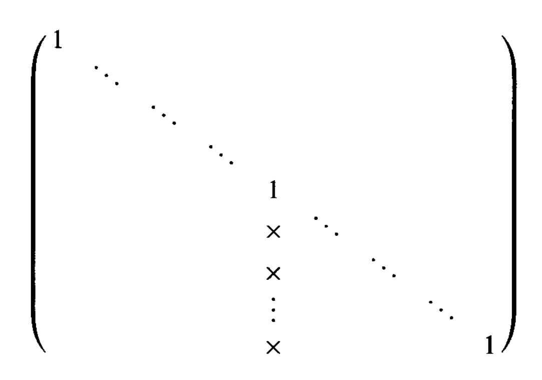
Givens旋转是具有如下形式的矩阵：

与单位阵的区别仅在$(i,i),(i,j),(j,i)$和$(j\ j)$四处。Givens旋转是正交的。

$$\begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \cos\theta & & \sin\theta & & \\ & & & \ddots & & & \\ & & -\sin\theta & & \cos\theta & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$$

Householder反射是由$\mathbf{I}-2\mathbf{w}^H\ \mathbf{w}$定义的矩阵，其中$\mathbf{w}^H\ \mathbf{w}=1$。这些矩阵是酉矩阵和Hermitian矩阵。

高斯变换矩阵有如下的形式：



其中×代表非零元素。它仅在对角线下方的一列上与单位阵有区别。

置换矩阵与单位矩阵有相同的列，但是次序不同。每行每列都有一个准确的单位元素。

B.6　向量和矩阵的范数

前面已经介绍了向量$\mathbf{x}$的**欧氏范数**或**2-范数**。

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2} = \sqrt{\mathbf{x}^H\mathbf{x}}.$$

介绍最大范数也是十分有用的。

$$\|\mathbf{x}\|_\infty = \max_{1\le i\le n} |x_i|.$$

另一个范数是1-**范数**。

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|.$$

所有这些都是更一般的***p*-范数**的特例。

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

范数常用来度量向量的大小或长度。

矩阵范数用下式来定义：

$$\|\mathbf{A}\| = \max_{\mathbf{x}\ne\mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|$$

可以得到下面的结论：

• A的1-范数是列和绝对值的最大值。

$$\|\mathbf{A}\|_1 = \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}|$$

• A的2-范数或谱范数：

$$\|\mathbf{A}\|_2 = \sqrt{\rho(\mathbf{A}^H\mathbf{A})}$$

而A的最大范数是行元素绝对值之和的最大值。

$$\|\mathbf{A}\|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}|.$$

另一个常用的矩阵范数是**F-范数**$\|\mathbf{A}\|_F$,用下面的式子定义：

$$\|\mathbf{A}\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2.$$

F-范数不能象另外三个矩阵范数那样由向量范数定义得到。

有下列不等式：

1) 对任何矩阵**A**和它所有的范数，有$\rho(\mathbf{A}) \leq \|\mathbf{A}\|$。

2) 如果**A**为Hermitian矩阵，则$\rho(\mathbf{A}) = \|\mathbf{A}\|_2$。

3) 如果**A**为酉矩阵，则$\|\mathbf{A}\|_2 = 1$。

使用矩阵范数可以对矩阵中扰动的灵敏性进行估计和度量。

线性方程组**Ax=b**的条件数定义为：

$$\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$$

有如下关系：

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \mathrm{cond}(\mathbf{A})\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}$$

其中，△**b**是对右侧**b**的扰动，而△**x**是对解向量**x**的相应的扰动。注意到关系中既有向量范数又有矩阵范数。对矩阵**A**中的扰动也有类似的关系。

B.7 矩阵因式分解

1) **LU因式分解或LU分解**　**PA=LU**，其中**P**为扰动矩阵，**L**为一个对角线上有元素为1的下三角矩阵，而U为一个上三角矩阵。

2) **Cholesky因式分解**　对称、正定矩阵**A**可以因式分解为$A=GG^T$，这里**G**为下三角阵。

3) **QR因式分解**　**A=QR**，其中**A**是$m\times n$矩阵，**Q**是一个$m\times m$的正交矩阵，而**R**是$m\times m$的上三角矩阵。

4) 假设A有n个线性无关的特征向量，则存在矩阵 **C**，使得$\mathbf{C}^{-1}\mathbf{AC}=\mathbf{D}$为对角阵，即$\mathbf{A}=\mathbf{CDC}^{-1}$，该条件是充分必要的。一个充分条件是所有的特征值均不同。

5) **舒尔分解**　对每个矩阵**A**，存在矩阵**U**，使得$\mathbf{U}^H\mathbf{AU}=\mathbf{T}$为上三角阵，即$\mathbf{A}=\mathbf{UTU}^H$。

6) 对于Hermitian矩阵**A**，存在一个酉矩阵**U**，使得$\mathbf{U}^H\mathbf{AU}=\mathbf{D}$为对角阵，即$\mathbf{A}=\mathbf{UTU}^H$。

7) **Murnaghan-Winters定理**　对于所有的实数阵**A**，存在实数正交阵**U**，使得$\mathbf{U}^T\mathbf{AU}=\mathbf{B}$为实数分块三角阵，这里对角线上的块为$2\times 2$阶或$1\times 1$阶。每个2阶块代表一个特征值的复数共轭对。

8) **Jordan标准形**　对每一个方阵**A**，存在一个非奇异矩阵**S**，使得$\mathbf{S}^{-1}\mathbf{AS}=\mathbf{J}$，其中**J**为块对角阵的形式：

如果块J_k为一阶，则$\mathbf{J}_k=(\lambda_k)$。与对角线上的每一个块相对应的一个特征向量也称为Jordan框。如果Jordan框的数目为p，则矩阵**A**有p个线性无关的特征向量。

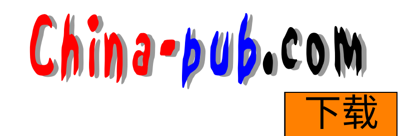
$$\mathbf{J}=\begin{pmatrix}\mathbf{J}_1 & & & & \\ & \mathbf{J}_2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \mathbf{J}_p\end{pmatrix} \qquad \mathbf{J}_k=\begin{pmatrix}\lambda_k & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & \ddots \\ & & & \ddots & 1 \\ & & & & \lambda_k\end{pmatrix}$$

9) **奇异值分解**　每个$m\times n$矩阵**A**都可分解成两个酉矩阵**U**和**V**，使得$\mathbf{U}^T\mathbf{AV}=\mathbf{D}$是一个$m\times n$对角阵。其中，**U**是一个$m\times m$矩阵，**V**是一个$n\times n$矩阵，而**D**的对角元素为$\sigma_k$。这些元素有序排列为$\sigma_1\geq\sigma_2\geq...\sigma_p\geq 0$，其中，$p\leq\min(m, n)$。如果有其他的$\sigma_k$，则为零。$\sigma_k$的值称为**A**的奇异值。因此$\mathbf{A}=\mathbf{UDV}^T$。

奇异值可用于定义**D**的广义逆$\mathbf{D}^+$。下面在$m \geq n$的情况给出的定义：

$$\mathbf{D} = \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_p & \\ \hline & 0 & & 0 \end{array}\right] \qquad \mathbf{D}^+ = \left[\begin{array}{ccc|c} \sigma_1^{-1} & & & \\ & \ddots & & 0 \\ & & \sigma_p^{-1} & \\ \hline & 0 & & 0 \end{array}\right]$$

但也同样适用于$m \leq n$。如果**D**为$m \times n$矩阵，则$\mathbf{D}^+$为$n \times m$矩阵。**A**的广义逆$\mathbf{A}^+$的大小也是$n \times m$，并且用奇异值分解定义为$\mathbf{A}^+ = \mathbf{V}\,\mathbf{D}^+\mathbf{U}^T$。

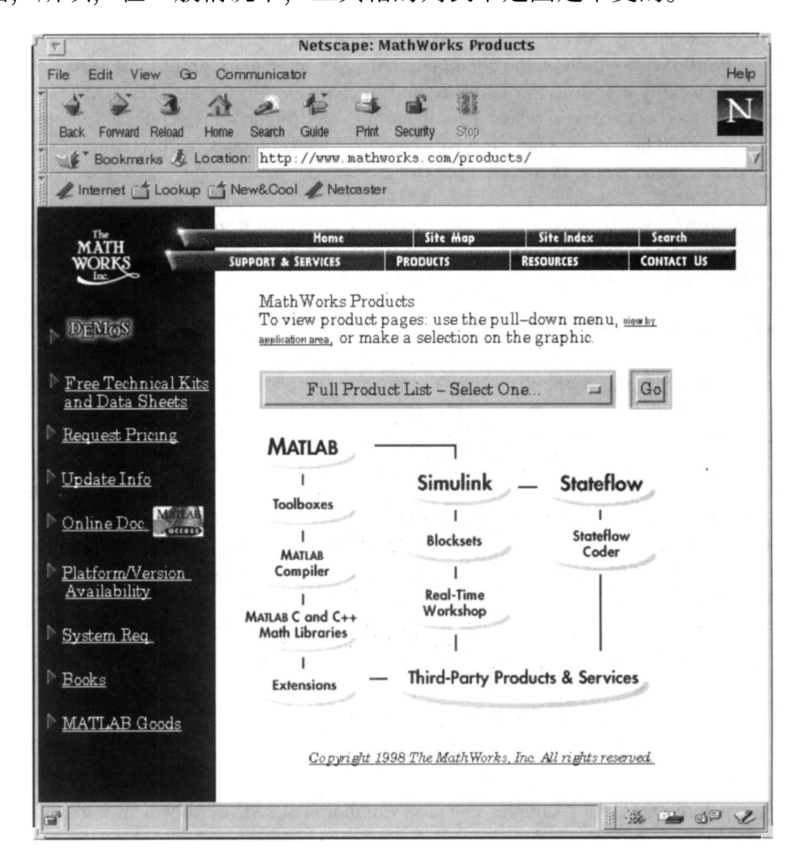


附录C　MATLAB工具箱和SIMULINK

这些信息是由MathWorks公司提供的，编辑成与MATLAB 5手册最后部分相一致的表格。

MATLAB拥有一个专用的家族产品，用于解决不同领域的问题，比如：信号分析、系统识别和仿真等。这些所谓的工具箱都用于 MATLAB的计算和画图。通常是 M文件和高级MATLAB语言的集合，以使得用户可以方便地修改函数的原代码，或增加新的函数。用户可以很方便地结合使用不同工具箱中的技术来设计针对某个问题的用户解决方案。

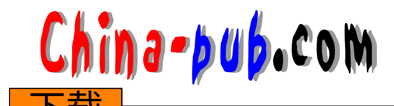
现在可以从MathWorks公司买到在图C-1中列出和描述的工具箱。由于每年都会开发出一些新的工具箱，所以，在一般情况下，工具箱的列表不是固定不变的。



图C-1　http://www.mathworks.com/products/

MATLAB 5的学生版可以处理元素最多为 16 384个的矩阵，还包括有 3个完全的工具箱：信号处理工具箱、控制系统工具箱和符号数学工具箱；参见表 C-1。可以注意到MATLAB学生版并没有设计专业工具箱供使用。学生版是由 Prentice Hall出版社出版发行的。

此外，还有一个功能强大的、可视化的、交互环境的工具 **SIMULINK**，用于模拟非线性



动态系统。SIMULINK提供一个用于创建动态系统对角模块的图形用户界面。由于SIMULINK充分利用了窗口技术，用户可以很容易地创建线性的、非线性的、离散的、连续的和混合模型。由于点击—拖动操作和鼠标交互的使用，来自块库的组件可以相互连结使用。在做‘what if’分析的过程中，可以改变参数。SIMULINK与MATLAB充分集成，与MATLAB和MATLAB工具箱一起使用，用户可以在建模、设计、分析和仿真的不同阶段之间移动。

SIMULINK是可以进行扩展的。该环境中包含了可选工具集，如：提高仿真速度，如表C-2所示。与SIMULINK相联系的工具箱成为块集，块集扩展了有专门设计和分析能力的块库；参见表C-3。

FEMLAB是用来解决偏微分方程（PDE)的另一个基于MATLAB的软件；参见http://www.femlab.com可以获得更多的信息。

作为一个相对较新的产品，**Stateflow**是一个功能十分强大的交互式设计和开发工具，可用于解决各种复杂的控制问题。Stateflow和SIMULINK以及SIMULINK的块扩展一起组成了用于解决事件控制和连续系统独有的环境。

使用MATLAB指令demo、helpdesk或浏览MathWorks公司的万维网主页：

http://www.mathworks.com/

可以获得关于SIMULINK、Stateflow和工具箱的更多信息。

此外，MathWorks公司提供匿名FTP服务器的文档服务。

该站点收集了与书中相关的M文件，以及用户和MathWorks公司提供的软件和文档。

ftp.mathworks.com

其他软件包括**MATLAB编译器**、**MATLAB C**和**C++数学库**，都可以从MathWorks公司买到。另外还有**Applix Link**和**Excel Link**，它们将字处理器和电子数据表格与MATLAB集成在一起，比如：将数值数据从MATLAB中移到Excel中。更多的信息可以从下面的站点获得：

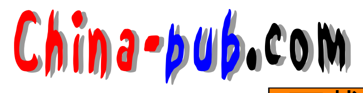
http://www.mathworks.com/products/

此外，还有第三个附加工具箱，**MATLAB Third Party products**，它使得MATLAB和SIMULINK更加完备。这些产品来自全世界各组织团体之间的合作。更多的信息可以从下面地址获得：

http://www.mathworks.com/connections

表C-1　MATHWORKS**公司开发的**MATLAB**工具箱**

‘μ’-*Analysis*和*Synthesis*	将MATLAB以及信号处理工具箱用于线性控制系统的健壮性分析和设计。
Chemometrics	用化学方法和技术对数据进行定量和定性的分析。
Communications	使用MATLAB函数和SIMULINK块进行通讯系统的设计、仿真和分析。
Control System	用于自动控制系统的设计和分析。该工具箱的函数包含在MATLAB学生版的信号和系统工具箱中。
Extended Symbolic Math	用于扩展的符号数学。增加对在Maple V中编程和访问



	所有Maple V库的支持。包括符号数学工具箱。
Financial	用于经济和定量的分析。
Frequency Domain System Identification	用于带或不带延时的线性系统的精确模型。基于频域数据。
Fuzzy Logic	用于模糊逻辑模拟智能控制产品和过程的开发。专门设计用于与SIMULINK一起工作。
Higher-Order Spectral Analysis	用于高阶光谱的高级信号处理。
Image processing	将MATLAB和信号处理工具箱一起用于图像和二维信号的高级处理和分析。
LMI Control	用于更快更有效地解线性矩阵不等式 (LMI)。
Mapping	用于分析和映射基于地理的数据。
Model Predictive Control	用于对包括操作和 /或控制变量约束的控制系统的设计和应用。
NAG Foundation	提供对NAG基本库中数学和统计事务的交互式访问。
Neural Network	用于各种神经网络和调节系统的设计、执行和仿真。包括SIMULINK扩展块库的附加块。
Optimization	用于线性和非线性函数的优化。
Partial Differential Equation (PDE)	用于实时的、二维空间且使用有限元素的方法研究和解偏微分方程。
QFT Control Design	与MATLAB和控制系统工具箱一起，使用 QFT方法进行实际设计健壮的反馈系统。
Robust Control	将MATLAB和控制系统工具箱一起用于高级的、健壮的和多变量反馈控制系统的设计。
Signal Processing	用于算法设计、数字信号处理和时序分析。
Spline	用于分段多项式函数、曲线拟合和函数逼近的构造和使用。
Statistics	用于统计数据分析、建模和蒙特卡洛仿真。同时提供概率统计基本概念的 GUI(图形用户界面)工具，并且建立用于创建个人统计工具的块。
Symbolic Math	用于符号数学、解方程、可变精度代数和专有的数学函数。该软件基于Maple V。MATLAB的学生版中包含了教育版。
System Identification	用于高级信号处理和建模，比如：参数建模、系统辨识和时序分析。建议使用信号处理工具箱。
Wavelet	用于信号和图像分析、压缩和降噪。



表C-2　SIMULINK的可选工具

SIMULINK	用于对直接来自SIMULINK块图的实时操作，自动生成C代码。
Real-Time Workshop	用于自动用Stateflow生成的SIMULINK模型部分的C代码。
Stateflow Coder	与SIMULINK、SIMULINK实时工作室的C代码生成器连结。

表C-3　SIMULINK块集

DSP Blockset	将MATLAB、SIMULINK和信号处理工具箱一起用于对SIMULINK和数字设计的实时工作室的使用扩展。
Fixed-Point Blocksets	用于定点应用的SIMULINK块库的扩展。比如：8位、16位、32位定点结果的选择。
Non-linear Control	将MATLAB和SIMULINK一起用于基于时域的控制设计。
Design(NCD)	包括SIMULINK扩展块库的附加块。
Power System Blockset	使用SIMULINK对电力网系统进行仿真。

在该手册中，有一些用PDE工具箱生成的图片。参见第1章的图1-11和图1-12。



附录D 快 速 参 照

这部分主要是《MATLAB指南》中命令的简短说明，供查阅使用。

编辑和特殊关键字

有些特殊关键字和系统有关。通常后面跟着‘ or’表示不同系统中的关键字。有的关键字就根本不在某些系统中存在。

↑或Ctrl-P↓	浏览并重新调用
或Ctrl-N	前面用过的命令
←或Ctrl-B	向左移动一个字符
→或Ctrl-F	向右移动一个字符
Ctrl-L或Ctrl-←	向左移动一个字
Ctrl-R或Ctrl-→	向右移动一个字
Ctrl-A或Home	移到一行的开始
Delete或Backspace	删除字符
Ctrl-K	删除到行末
Ctrl-C	停止运行计算
cedit	切换键模式

基本的系统命令

exit, quit	退出MATLAB
diary	文本记录
save	将工作区保存到文件中
load	从文件中装载工作区
type,dbtype	列文件清单
what,dir,ls	列出目录中的内容
cd	改变目录
pwd	显示当前目录
path	显示并设置当前路径
!	后面跟操作系统命令

帮助和演示命令

在Macintosh和Windows版本中，帮助可以很容易地从帮助菜单中获得。

help	关于某一主题的帮助
Lookfor	查找文本
expo, demo	演示程序
whatsnew	列出新的特征
info	一般信息

变量和工作区

who, whos	列出变量
clear	清除变量
size, length	矩阵和向量的大小
exist	存在性
pack	重构工作区
format	输出格式
casesen	切换字母大小写

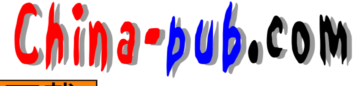
标准变量和常量

ans	上一个无符号的答案
pi	π, 3 .141 592 6535
eps	相对精度
realmax,realmin	最大数和最小数
inf	无穷大，定义为1/0
NaN	不是数字，如：0/0
i,j	虚部，$\sqrt{-1}$
Nargin,nargout	参数个数

User I/O

获取一个变量值最简单的方法是键入变量名并按回车键。

disp	显示值或文本
input	从键盘输入
ginput	读入坐标
pause	暂停执行
waitforbuttonpress	等待用户指令
format	输出格式
more	滚动格式


下载

Casesen 切换字母大小写
menu 可以选择的弹出式菜单
lasterr 上一次的错误信息字符串

要了解图形命令和图形用户界面的有关信息，可参见下面的图形和句柄图形一节。

计时函数

flops 浮点运算的次数
tic,toc,etime 计时
clock,date 时间和日期
cputime MATLAB运行的时间

特殊的系统命令

computer,getenv 计算机类型
terminal 设置终端类型
ver 版本信息等
version MATLAB版本
hostid 服务器主机号

数学函数

用标准的数学函数来实现元素操作。

基本的数学函数

abs 绝对值
sign 符号函数
sqrt 平方根
pow2 2的乘幂
exp 指数函数
log,log2,log10 对数函数
sin,cos,tan,cot, 三角函数
sec,csc
asin,acos,atan2, 反三角函数
atan,acot,asec,acsc 双曲函数和反双曲函数
sinh,cosh,tanh,
coth,asinh,acosh,
atanh,acoth,sech,
csch,asech,acsch

高级数学函数

legendre 勒让德函数
bessel,bessely 贝塞尔函数
gamma,gammaln, 伽马函数
gammainc
beta,betaln,betainc Beta函数
expint 指数形积分
erf,erfinv,erfc,erfcx 误差函数
ellipke,ellipj 椭圆积分

坐标变换

cart2pol,pol2cart 笛卡儿坐标和极坐标
cart2sph,sph2cart 笛卡儿坐标和球形坐标

整数和符点数

round,fix,floor,ceil 舍入函数
rat 有理数近似
rats 有理数转化成字符串
rem 除法所得的余数
gcd 最大公约数
lcm 最小公倍数

复数

real,imag 实数和虚数部分
conj 共轭
angle 相位角
unwrap 调整参数
cplxpair 复数对

矩阵运算和函数

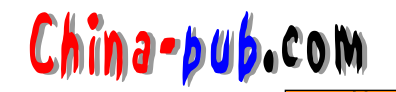
操作符前的点表示一个元素操作运算。

矩阵运算符

+, — 加法和减法
, .,cross,dot, 乘法
kron
/,\,./,.\ 除法
‘,.’ 共轭，转置
^,.^ 乘幂
>,<,>=,<=,==,~= 关系运算符
&,|,~,xor 逻辑运算符

矩阵函数

det,trace,rank 行列式、迹和秩



inv,pinv	逆和伪逆
orth,null	基本子空间
subspace	子空间之间的夹角
expm,logm,sqrtm, funm,polyvalm	矩阵函数
size,length	矩阵和向量的大小和长度
any,all,isnan, isinf,isieee, issparse,isstr, isempty,finite	逻辑函数
find	按条件查找

定义向量和矩阵

冒号“ :”用来产生和抽取向量和矩阵。

:	下标运算符
linspace,logspace	产生向量
eye	单位矩阵
ones,zeros	1矩阵和零矩阵
rand,randn	随机矩阵
diag	对角阵
triu,tril	三角阵
fliplr,flipud, rot90,reshape	变换矩阵
hilb,invhilb, toeplitz,compan, gallery,hadamard, hankel,magic, pascal,rosser, vander,wilkinson	特殊矩阵

字符串

字符串是用单引号括起来的，如‘ text ’。

strcmp	字符串比较
strtok, strrep	抽取字符串
findstr	查找字符串
isstr,isletter,isspace	字符串逻辑
strmat	字符串矩阵
blanks,deblank	字符串中的空格
lower,upper,abs	小写、大写和 ASCII 码转换
setstr,num2str, int2str, rats,hex2num, hex2dec,dec2hex	字符串转换
sprintf,sscanf	格式化输入输出
eval,feval	字符串求值

数据分析和统计

sum,cumsum	求和
prod,cumprod	乘积
diff,gradient,del2	差分
max,min	最大值和最小值
mean,median	平均值和中位值
std	标准偏差
cov	方差和协方差
corrcoef	相关矩阵
sort	排序
hist,bar,stairs	统计频数直方图等

线性系统

线性方程组通常用反斜杠运算符“\”求解。

\	左除，解操作符
det,rank	行列式和秩
inv	矩阵的逆
norm,normest	矩阵范数
cond,condest	条件数
lu	LU分解
rref,rrefmovie	矩阵的Echelon形式
chol	Cholesky因式分解
qr	QR因式分解
qrinsert,qrdelete	QR操作
planerot	Givens旋转

特征值和特征向量

eig,polyeig	特征值和特征向量
poly	字符型多项式
trace	矩阵的迹
balance	平衡变换
hess	上海森伯形式
qr,qz	QR和QZ分解
schur	舒尔分解
rsf2csf	实数到复数的舒尔形

	式转换
cdf2rdf	复数到实数的对角形式转换
svd	奇异值分解

稀疏矩阵

大多数标准矩阵命令可以直接用于稀疏矩阵。最明显的例外是norm命令和一些图形命令。

sparse	将满矩阵变换成稀疏矩阵
full	将稀疏矩阵变换成满矩阵
find	查找下标
spconvert	用下标构成稀疏矩阵
nnz	非零元素的个数
spy	结构绘图
nonzeros	查找非零元素
speye	稀疏单位阵
spones	非零位置上的1矩阵
sprandn,sprandsym	稀疏随机矩阵
spdiags	稀疏对角阵
issparse	存储的逻辑值
spalloc, nzmax	稀疏矩阵的分配
spfun	函数求值
sprank	稀疏矩阵的秩
normest	2-范数估计值
condest	条件数估计值
spaugment	创建平方矩阵以计算最小的平方解
etree	矩阵的消元树
etreeplot	消元树图形
colmmd,symmmd	最小度次序
symrcm,colperm	列置换
randperm	序列变换向量
dmperm	Dulmage-Mendelsohn 分解
spparms	设定稀疏参数
symbfact	分析chol和lu
gplot	图形绘制

多项式和曲线拟合

polyval, polyvalm	多项式求值
conv, deconv	卷积，多项式乘积
residue	部分分数
polyder	多项式导数
poly	字符型多项式
compan	伴随矩阵
polyfit	多项式的近似值
interp1-interp6	插值
interpft	傅里叶插值
spline	三次样条插值
legendre,bessel, bessely	正交函数
fft,ifft	快速傅里叶变换
fft2,ifft2	DFFT
fftshift	交换象限

零值，最大值和最小值

roots	多项式的零值
fzero	函数的零值
fmin,fmins	函数的最小值

积分和微分方程

trapz,quad,quad8	定积分计算
ode23,ode45,ode23p	ODE解法

MATLAB程序设计

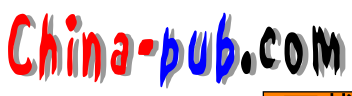
可以在提示符下编写程序，或更方便的方法是在M文件中编写程序。 M文件是由MATLAB命令组成的扩展名为.m的文件。M文件通过给出文件名作为命令来执行。MATLAB中的块用end结束。

条件语句

if条件 语句1 语句2 ……	if-else块的一般格式
else 语句3 end	else部分是可选的，但是,if总是有一个end

循环

fori=1:2:10 语句1 ……	i从1到10循环，步长为2



end	“块”以end结尾
while条件	当条件为真时，循环
语句1	
……	
end	

控制语句

%	后面跟注释
return	退出M文件
pause	暂停运行
break	终止当前循环
global	定义全局变量
nargin	输入的参数个数
nargout	输出的参数个数

调试M文件

keyboard	键盘命令模式
echo	Echo命令
error	终止并显示错误信息
dbtype	显示带有行号的M文件
dbstop,dbclear	设定和清除断点
dbstatus	列出当前断点
dbstep,dbcont	执行语句
dbup,dbdown	切换工作区
dbstack	显示工作区栈
dbquit	退出调试模式

图形

二维和三维图形

plot	绘制二维图形
plot3	绘制三维图形
fplot	绘图函数
subplot	将当前图分为若干子图
errorbar	绘制带误差的条形图
comet,comet3	动态地绘制二维、三维图形
polar	在极坐标中绘图
semilogx,semilogy loglog	对数绘图
quiver,feather, compass,rose	复数图形
stem	数据序列图
hist,bar,stairs	统计频数直方图等

图形控制

figure	创建和显示图形
clf	清除图形
hold	保留当前图形
subplot	将当前图形分为若干子图
clc	清除命令窗口
home	将光标移至开始，如：左上角
axis	坐标轴刻度
zoom	放大或缩小（仅限二维）
grid	显示或隐藏网格线
title,xlabel,ylabel, zlabel	给出基本的文本项
text	在某处写文本
gtext	用鼠标放置文本
ginput	读坐标
rbbox	移动矩形区域
hidden	显示或不显示隐藏的曲面图
view	观察点的位置或角度
viewmtx	定义观察点的矩阵
rot90	旋转矩阵

曲面图和等高线图

contour	等高线图
contour3	三维等高线图
clabel	标记等高线
meshgrid	产生网格
cylinder,sphere	特殊的几何网格
surf	曲面图
mesh	网格曲面图
meshc,meshz, waterfall	带有参考线的网格曲面图
surfl,surfc, surfnorm	带有特殊光照，等高线和法线的曲面图
pcolor	俯视的曲面图
fill,fill3	填充多边形



slice 三个变量的绘图函数

颜色控制

shading 曲面图颜色模式
colormap 读或设置颜色表
colorbar 显示颜色条
rgb2hsv,hsv2rgb 颜色表的旋转
caxis 颜色坐标轴的刻度
spinmap 旋转颜色
brighten 改变颜色映射
contrast 增加对比度
whitebg 背景颜色
graymon 黑色或白色参数

打印

print 生成硬拷贝图形
printopt 打印选项
orient 纸的方向

声音

sound 播放声音
saxis 声音坐标轴
auread,auwrite Sun工作站上的声音文件
mu2lin,lin2mu Sun工作站上的声音转换
wavwrite,wavread Windows声音文件

句柄图形

MATLAB中的图形是面向对象的。首先有一个根对象，它可能有许多子对象，即图形。这些图形可以包含一个或更多的轴、绘图区域。一个图形是由绘制在图形区域中的线条、曲面、补片以及文本对象构成的。每个对象有一个唯一的句柄，通过该句柄可以改变该对象的属性。

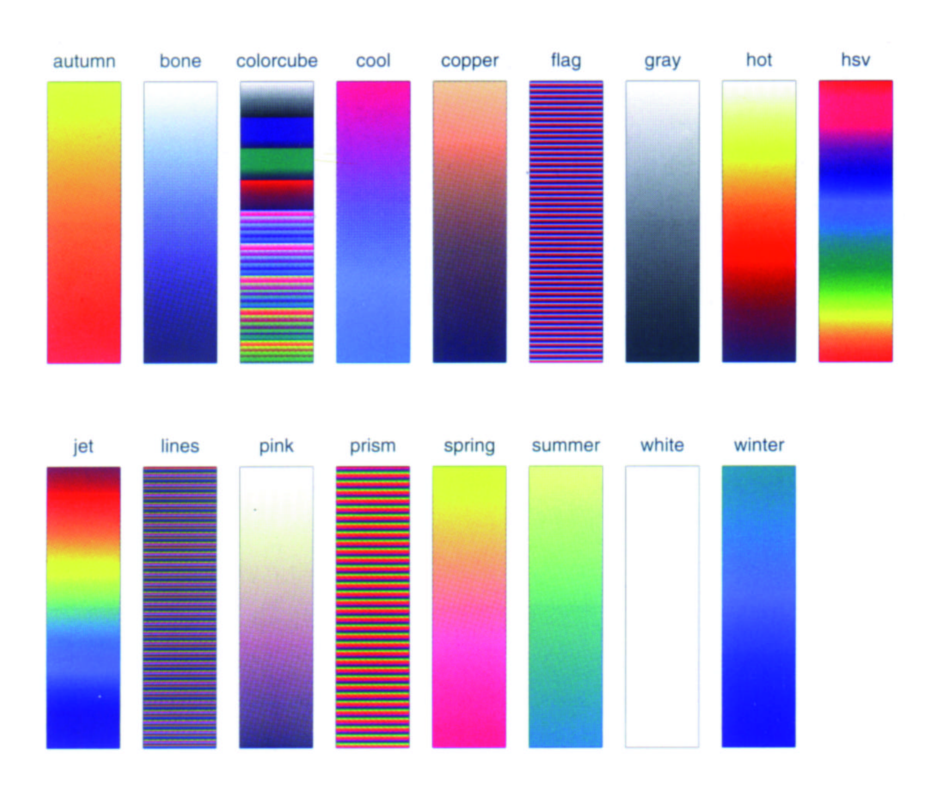
get 获得属性
set 设置属性
gcf,gca,gco 获得当前图、坐标轴或对象的句柄
clf,cla 清除当前图形或坐标轴
close 关闭图形
delete 删除对象
rotate 旋转对象
reset 恢复某一对象的属性
refresh 刷新图形
drawnow 更新图形
newplot 设定下一个图的属性
figure 设为当前图或创建图形
axes 绘图区域
line 线对象
text 文本对象
patch 补片填充的多边形对象
surface 曲面对象
image 图像对象
capture 位图拷贝
uimenu 用户界面菜单
dialog 对话框
errordlg,warndlg,helpdlg,questdlg 继承的对话框

动画

movie 放映动画
getframe 获取动画帧
moviein 初始化动画

二进制和文本文件

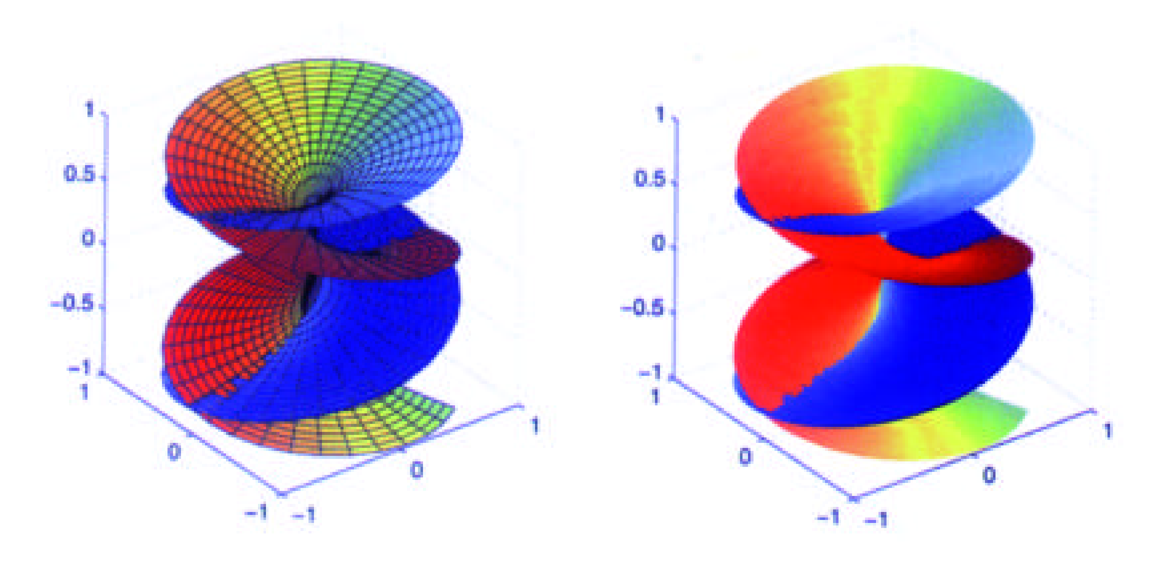
fopen 打开文件
fclose 关闭文件
fwrite 写文件
fread 读文件
fprintf 格式化输出到文件
fscanf 从文件读数据
fegtl,fgets 从文件读取一行数据
ferror 检查文件错误
feof 检查文件结尾
frewind 重新设置文件
fseek 在文件中设定位置
ftell 从文件中获取位置



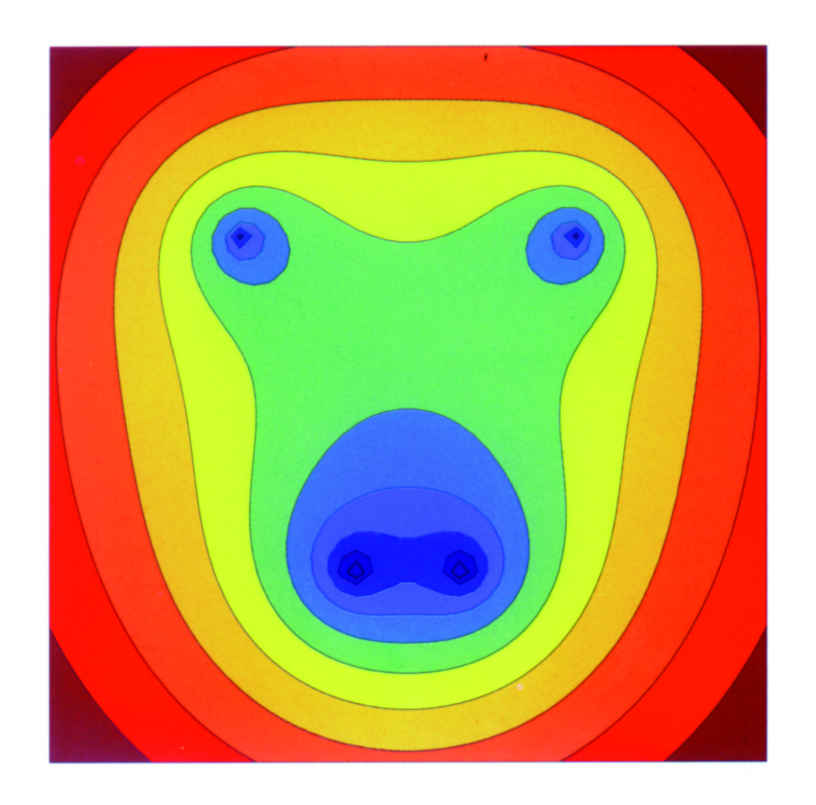
图P-1　MATLAB中预定义的17种不同的颜色图案(定义在13.6节的表13-2中)。使用的命令有：*colormap* 、*pcolor* 和*subplot* 。注意，命令*contrast* 也可以针对某幅图创建一个用户自定义颜色图案



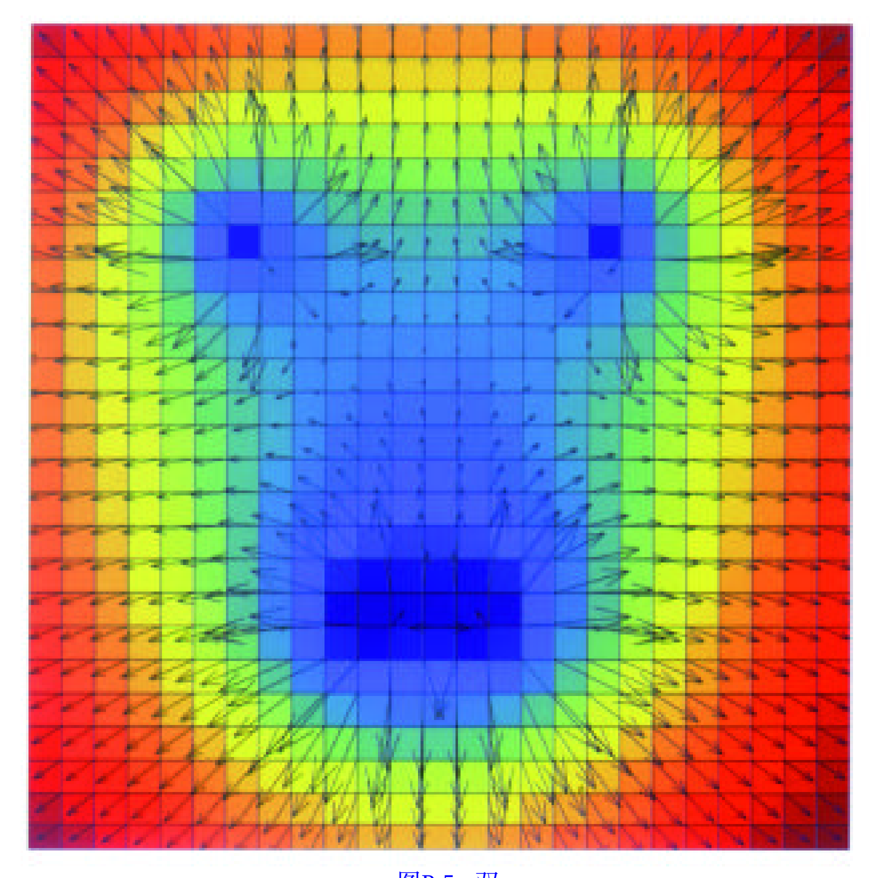
图P-2　这个卡通几何物体可以用命令*sphere* 、*mesh* 、*light* 、*view* 、*axis* 和图形句柄来创建



图P-3　第3.75个根的黎曼表面图可以用命令*cplxroot*、*subplot*、*shading faceted*(左)和*shading interp*(右)画出

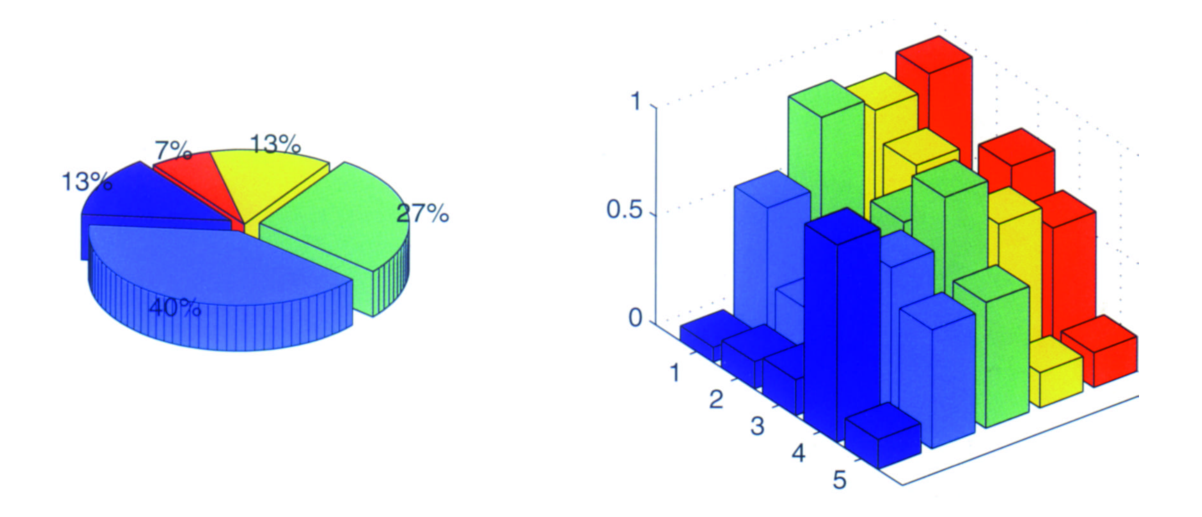


图P-4　双变量函数 $z=f(x, y)$的等高线图可以用命令*contourf* 画出。在等高曲线z之间的区域用颜色填充，用到的相关命令为：*meshgrid*、*colormap* 和*axis*



图P-5　双

变量函数 $z=f(x, y)$的梯度向量图形可以用命令*meshgrid* 、



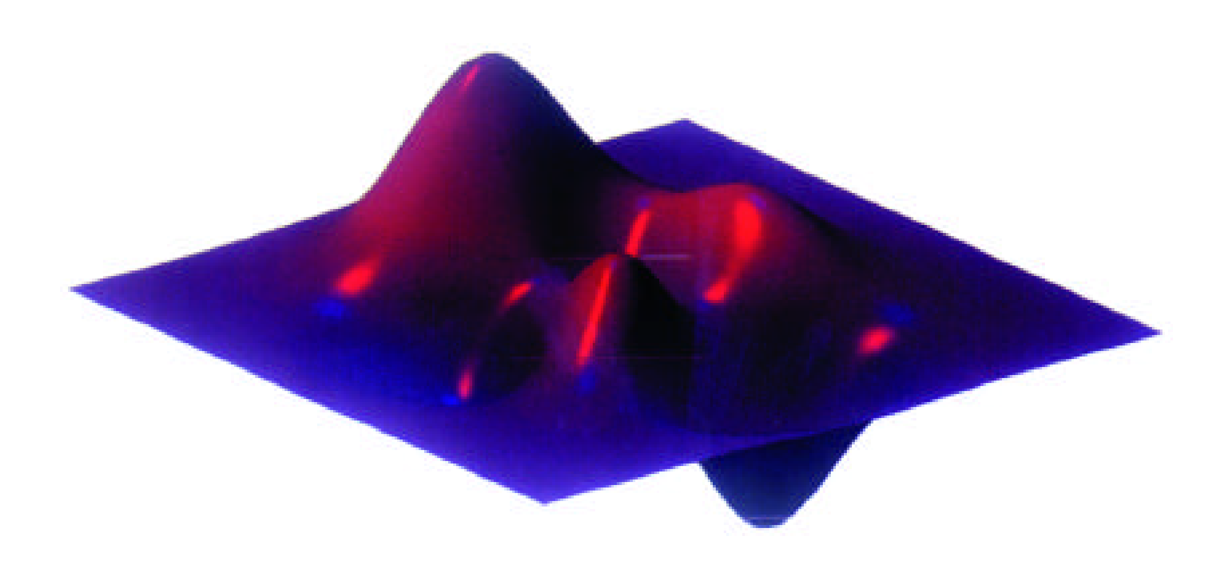
图P-6　命令*subplot* 可以画出3D的饼图(左)和直方图(右)。使用的相关命令为：

pcolor 、*hold* 、*gradient* 和*quiver* 画出

pie3 、*bar3* 、*colormap* 和*caxis*



图P-7　MATLAB中可以显示图像。使用的命令为：
load earth、*image*、*colormap* 和*axis*



图P-8　双变量函数 $z=f(x, y)$的表面图使用颜色句柄和光对象。这里用了一个红色和一个蓝色光源。相关的命令为：*meshgrid*、*peaks*、*surf*、*colormap*、*material* 和*light*



参 考 文 献

本参考文献包括有关线性代数、矩阵代数和应用程序的书。还列出了基本的MATLAB手册。

Backstrom, G. *Practical Mathematics Using MATLAB*. Studentlitteratur (ISBN 91-44-49231-6) and Chartwell Bratt Ltd (ISBN 0-86238-397-8), 1995.

Biran, A. and Breiner, M. G. *MATLAB for Engineers*, Addison-Wesley, 1995 (ISBN 0-201-56524-2).

Borse, G. J. *Numerical Methods with MATLAB: A resource for Scientists and Engineers*, PWS Publishing Company, 1997 (ISBN 0-534-92822-1).

Golub, G. H. and Van Loan, C. F. *Matrix Computations* (2nd edition). The Johns Hopkins Univeristy Press, 1989 (ISBN 0-8018-3739-1).

Hager, W. W. *Applied Numerical Linear Algebra*. Prentice Hall, 1988 (ISBN 0-13-041369-0).

Heath, M. T. *Scientific Computing, an Introductory Survey*, McGraw-Hill, 1997 (ISBN 0-07-027684-6).

Jennings, A. and McKeown, J. J. *Matrix Computation*. John Wiley, 1992 (ISBN 0-471-93527-1).

Lindfield, G. and Penny, J. *Numerical Methods Using MATLAB*. Ellis Horwood, 1995 (ISBN 0-13-030966-4).

Malek-Madani, R. *Advanced Engineering Mathematics with Mathematica and MATLAB*, Addison-Wesley, 1998 (ISBN 0-201-59881-71).

Marcus, M. *Matrices and MATLAB: A Tutorial*. Prentice Hall, 1993 (ISBN 0-13-562901-2).

Ogata, K. *Solving Control Engineering Problems with MATLAB*. Prentice Hall, 1993 (ISBN 0-13-045907-0).

Strang, G. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 1993 (ISBN 0-9614088-5-5).

Strum, R. D. and Kirk D. E. *Contemporary Linear Systems Using MATLAB*. PWS Publishing, 1994 (ISBN 0-534-93273-8).

MATLAB手册：

Getting Started with MATLAB, The MathWorks, Inc., 1998.

Late-Breaking News for the MATLAB 5.2 Product Family, The MathWorks, Inc., 1998.

MATLAB 5.2 Product Family New Features, The MathWorks, Inc., 1998.

MATLAB Application Program Interface Guide, The MathWorks, Inc., 1998.

MATLAB Application Program Interface Reference Manual, The MathWorks, Inc., 1998.

MATLAB Functions Reference – Volume 1, The MathWorks, Inc., 1998.

MATLAB Functions Reference – Volume 2, The MathWorks, Inc., 1998.

MATLAB Application Program Interface Reference Manual, The MathWorks, Inc., 1998.

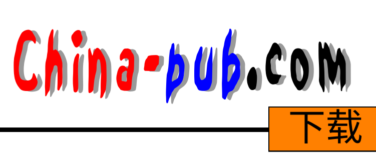
Using MATLAB, The MathWorks, Inc., 1998.

Using MATLAB Graphics, The MathWorks, Inc., 1997.

Student Edition of MATLAB Version 5 User's Guide, Prentice Hall, 1995 (ISBN 0-13-272550-9).



命令表清单


China-pub.com
下载